PHOTOGRAPH THIS SHEET

AD A093565

DTIC ACCESSION NUMBER

LEVEL II

INVENTORY

Integrated Sciences Corp.
Santa Monica, CA
"Dynamic Displays For Tactical Planning
Volume III: Software Documentation
Rpt. No. ARI-RN-80-9 Final Rpt. Dec. 79
Contract No. MDA903-78-C-2012 Rpt. No. ISC-281-4

DOCUMENT IDENTIFICATION

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DISTRIBUTION STATEMENT

ACCESSION FOR		
NTIS	GRA&I	☒
DTIC	TAB	☐
UNANNOUNCED		☐
JUSTIFICATION		
BY		
DISTRIBUTION /		
AVAILABILITY CODES		
DIST	AVAIL AND/OR SPECIAL	
A		

DISTRIBUTION STAMP

DTIC
ELECTE
S NOV 14 1980 D
D

DATE ACCESSIONED

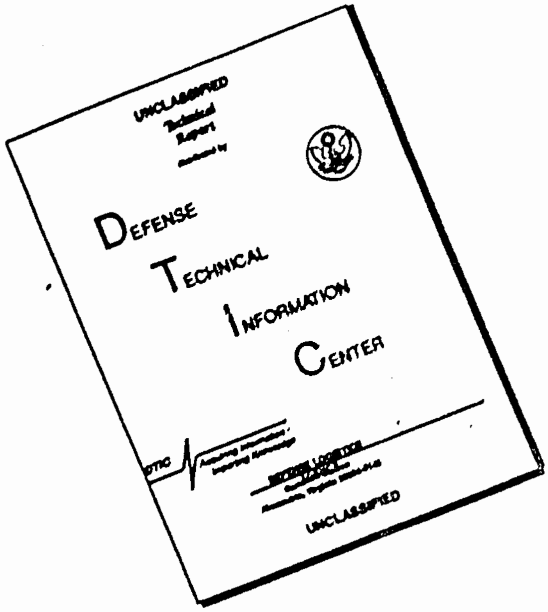
DATE RECEIVED IN DTIC

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-DDA-2

DTIC FORM OCT 79 70A

DOCUMENT PROCESSING SHEET

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

RESEARCH NOTE 80-9

Dynamic Displays For Tactical Planning
Volume III: Software Documentation

Michael D. Schechterman and Lawrence R. Levi
Integrated Sciences Corporation

Human Factors Technical Area

DECEMBER 1979

U. S. Army
Research Institute for the Behavioral and Social Sciences

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER Research Note 80-09	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Dynamic Displays for Tactical Planning Volume III: Software Documentation		5. TYPE OF REPORT & PERIOD COVERED Final: July 78 - Dec 79 6. PERFORMING ORG. REPORT NUMBER 281-4
7. AUTHOR(s) Michael D. Schechterman and Lawrence R. Levi		8. CONTRACT OR GRANT NUMBER(s) MDA 903-78-C-2012
9. PERFORMING ORGANIZATION NAME AND ADDRESS Integrated Sciences Corporation 1640 Fifth Street Santa Monica, CA 90401		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2Q162722A765
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Research Institute for the Behavioral and Social Sciences, 5001 Eisenhower Avenue Alexandria, VA 22333		12. REPORT DATE December 1979 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

The research was monitored technically by Franklin L. Moses of the Human Factors Technical Area, ARI.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Decision Aids	Situation Maps
Automated Displays	Graphic Displays
Command and Control	Interactive Graphics
Battlefield Displays	Dynamic Graphics
Tactical Planning	Battlefield Models

20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Volume III of the three-volume set contains detailed documentation of computer software used to support research on two-sided, user-controlled, dynamic wargaming. The current volume is intended for systems programmers/technical personnel who are interested in specifics of implementation. Complete program listings are provided. The project emphasizes that a battlefield planner can work in harmony with computer graphics to structure and analyze battlefield situations. Special tactical displays and dynamic replays of events are designed to aid the planner. (over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Abstract (cont)

Volume I, published as ARI Research Report 1247, provides a functional description of the research for Army managers, command staffs, and other potential users of the concepts. Volume II, ARI Technical Report 455, is intended for readers with specialized interests in research and development of interactive graphics for battlefield applications.

TABLE OF CONTENTS

LIST OF FIGURES

1.0 INTRODUCTION

This document is Volume III of the three-volume report on the Tactical On-Line Maneuver Model (TOMM). Its purpose is to document as completely as possible in a single volume the software for TOMM. It is intended for the technical reader interested in the details of system implementation.

The TOMM software consists of some 8400 lines of source code, with 1.5% written in assembly language and the rest in FORTRAN. This does not include the library of FORTRAN-callable graphics subroutines supplied by the vendor of the display system. TOMM is written for a minicomputer-based color refresh vector graphics system which resides at Integrated Sciences Corporation. It consists of a Varian V73 16-bit minicomputer with 32K words of memory and an IDIIOM II color vector display generator (stroke writer) with CPS monitor. For more information on the hardware, refer to Volume II, Section 3.0 of the TOMM report.

The remainder of Volume III is organized in the following way. Section 2 contains information on the overall organization of the TOMM software and the highest level control of program flow within TOMM. Section 3 contains descriptions of all the major data structures of TOMM and the flow of data into and out of these structures. Section 4 contains more detailed descriptions of the major software modules of TOMM. Finally, Section 5 is a complete source listing of the TOMM software.

2.0 SOFTWARE ARCHITECTURE AND CONTROL FLOW

2.1 TOMM SOFTWARE HIERARCHY

The TOMM software is organized as a multi-level tree hierarchy with a root segment at the highest level and twenty-one segments called overlays at the second level. Each overlay generally corresponds to a single major function or set of related functions. In some cases, due to the limited addressing capability of the 16-bit minicomputer on which TOMM was implemented, single functions were broken up into multiple overlays in order to stay within the available memory space. Only one overlay at a time resides in memory; the others reside on disk. The first two levels of TOMM are shown in Figure 2-1.

The root segment determines which overlay is to be executed next, and provides communication between overlays. The minicomputer operating system dictates that all communication between overlays must occur through common areas in the root segment. The root segment thus contains over eight thousand words of common area: two large files (the Master Data File and the display file) and a number of auxiliary variables. The structure and flow of data in TOMM will be described in Section 3.0 of this volume.

Each overlay in turn is at the top of a tree hierarchy containing an average of about fifteen modules, some of which are repeated in other overlays. Figure 2-2 shows the complete structure of one of the simpler overlays, namely overlay 3 which handles reference lines.

2.2 FLOW OF CONTROL

In the overlay structure described in the previous section, only the root segment can call overlays into main memory from disk and pass information between overlays. In order to conserve memory, however, the root segment contains only four executable statements. Actual control of program flow therefore resides in an overlay which acts as the "director." Overlay 1, called DRECTR, serves this function. Most of the sequencing of overlays for the performance of TOMM functions is controlled by DRECTR; most transitions between overlay modules of a single function pass through it.

First level:

Root segment : Calls all second level routines

Second level (overlays):

0 1 = DRECTR : The director (see Section 2.2)
0 2 = SCINIT : Initialize scenario display
0 3 = REFLIN : Reference lines
0 4 = DEFTER : Define terrain
0 5 = UNTMOB : Unit mobility illustrations
0 6 = UNITS : Establish initial order of battle
0 7 = USTATE : Add and delete units
0 8 = MVMMI
0 9 = UNTPTH
010 = TERPTH : Define single-unit and cluster movements
011 = UNTARL
012 = CLSARL
013 = AOBE1 : Draw AOBE's (with 017)
014 = RNGCNT
015 = URISE : Draw future position contours
016 = CONGEN
017 = DNCTR1 : Draw detection contours
018 = DNCTR2
019 = UUDCT : Compute unit-to-unit detections
020 = SETIME : Set time-of-day
021 = REPLAY : Replay scenario (With 019 for current replay only)

Figure 2-1. First Two Levels of TOMM Software Hierarchy.

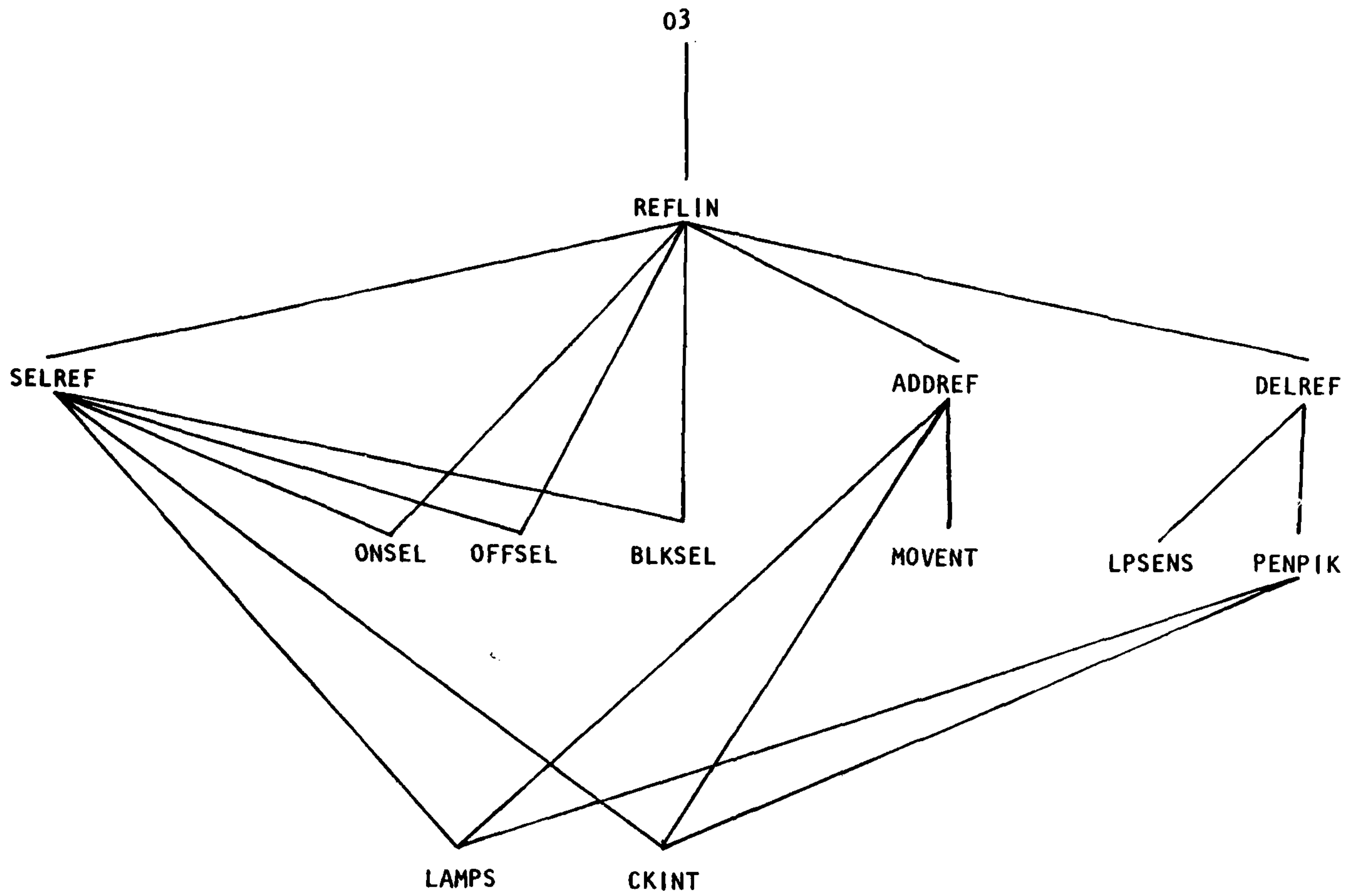


Figure 2-2. Detailed Structure of a Single Overlay.

In order to accomplish this control flow director function, DRECTR also serves as the interface with the user for the main functional menu of TOMM. It waits for the user to select the next function and sets up the sequence of overlays necessary to carry out that function. When the function sequence is complete, control returns to DRECTR. Figure 2-3 is a diagram showing the highest level flow of control in TOMM, i.e., at the overlay level. Functions not shown are performed within DRECTR itself.

DRECTR

↓↑

Root Segment

↓↑

Overlay sequences (control passes to DRECTR between overlays in the sequence, unless otherwise noted):

- Define terrain and initial order of battle : DEFTER, UNTMOB, UNITS
- Recall scenario : DRECTR itself, with UNTMOB
- Reference lines : REFLIN
- Unit state (add/delete units) : USTATE
- Future position contours : RNGCNT, URISE, RNGCNT, CONGEN, RNGCNT
- Master replay : REPLAY
- Current replay : UUDCT, REPLAY (control passes directly between these)
- Set time-of-day : SETIME
- Detection contours : DNCTRI, DNCTR2
- AOBE's : DNCTRI, AOBE1
- Unit movement definition : MVMMI, UNTPTH, TERPTH, UNTARL (single-unit)
  MVMMI, $\underbrace{\text{(UNTPTH, TERPTH)}}_{\text{Once for each cluster member}}$, CLSARL (cluster)

Figure 2-3. Highest-Level Flow of Control in TOMM.

3.0 DATA: STRUCTURES AND FLOW

3.1 DATA STRUCTURES

The major data structures of TOMM are the Master Data File (MDF) and the display file. The MDF is used to store almost all information about the current scenario. There are essentially four subfiles included in the MDF: terrain information, unit state information, unit actions (primarily movements), and events (detections and engagements). The MDF is described in detail in Section 3.1.1 below.

The display file is used to store display commands which give a picture of the scenario at the current problem time, together with all other information displayed to the used at any given time. It is described in detail in Section 3.1.2 below. Several other auxiliary data structures are discussed in Section 3.1.3.

3.1.1 Master Data File (MDF)

The Master Data File, an integer array of 3000 words with the variable name MDF, is the framework behind any scenario defined in TOMM. At the onset MDF holds all information about the terrain and initial order of battle, if one is defined. Should more units be added or deleted at a later time, the MDF is so changed. When unit movements are defined they are stored in this array, and as these movements are played out the MDF is continually updated to reflect each unit's new location and state (e.g., which units it detects, what terrain it occupies). As events (detections or end of detections) occur, these are stored in the MDF to be recalled when master replay is run.

The overall structure of the MDF is shown in Figure 3-1. The header is the first four words of the MDF and these are pointers to the areas in the array which contain terrain data, unit data, event data and scratch. For example, suppose the first four words of MDF were as follows:

MDF

1	5
2	996
3	2121
4	2100

The first integer, 5, is the terrain pointer and therefore, information concerning terrain begins at MDF(5), while unit data begins at MDF(996), the scratch area starts at MDF(2121), and MDF(2100) is the first word of the event area.

Each terrain feature of a scenario is allotted space (of varying length) in the MDF organized as follows (see Figure 3-2):

1. The first word is a pointer to the beginning of the next terrain stored in the MDF or 0, if it is the last terrain feature.

2. The second word is the code for the terrain type (1 = road, 2 = river, 4 = lake, 8 = city, 16 = outer hill, 32 = forest, 64 = simple road, and 128 = inner hill).

3. The third word is the entity number of this feature.

4. The following four words are Xmin, Ymin, Xmax, Ymax which denote the smallest X value, smallest y value, largest X value, and the largest Y value of the given terrain.

5. The last block of words detailing a terrain feature are the sequence of points X_1, Y_1, X_2, Y_2, ...X_n, Y_n which delineate the region. (the number n may be different for each feature). Except for simple roads and rivers all terrains are closed contours and therefore, $(X_1, Y_1) = (X_n, Y_n)$. Roads are stored twice, once as a number of line segments (simple road) and once as a narrow closed contour which encloses the simple road. Only the latter are displayed.

The next area, beginning at MDF (MDF(2)), holds data concerning the current status of each unit. The header of this area consists of four

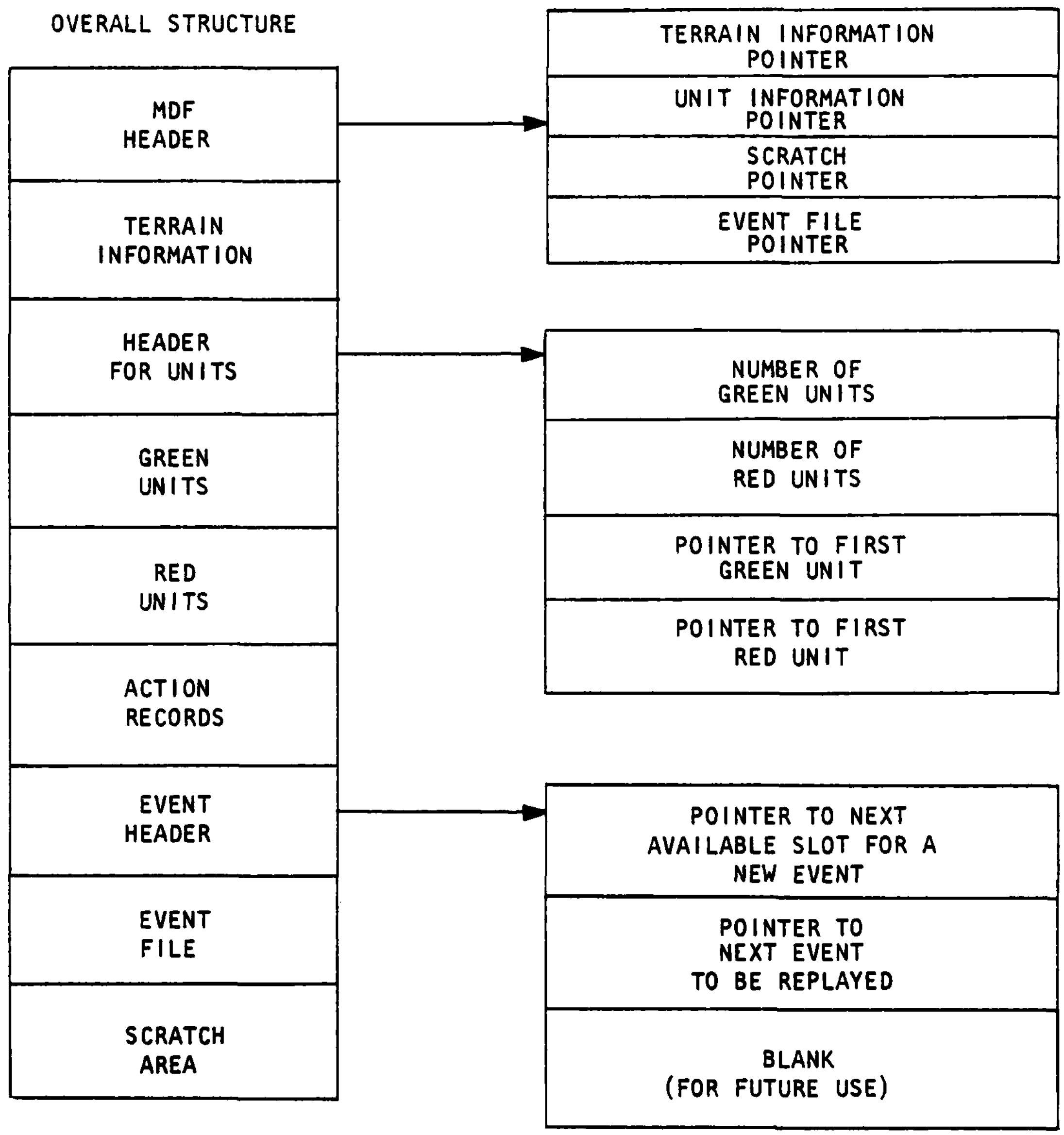


Figure 3-1. Master Data File.

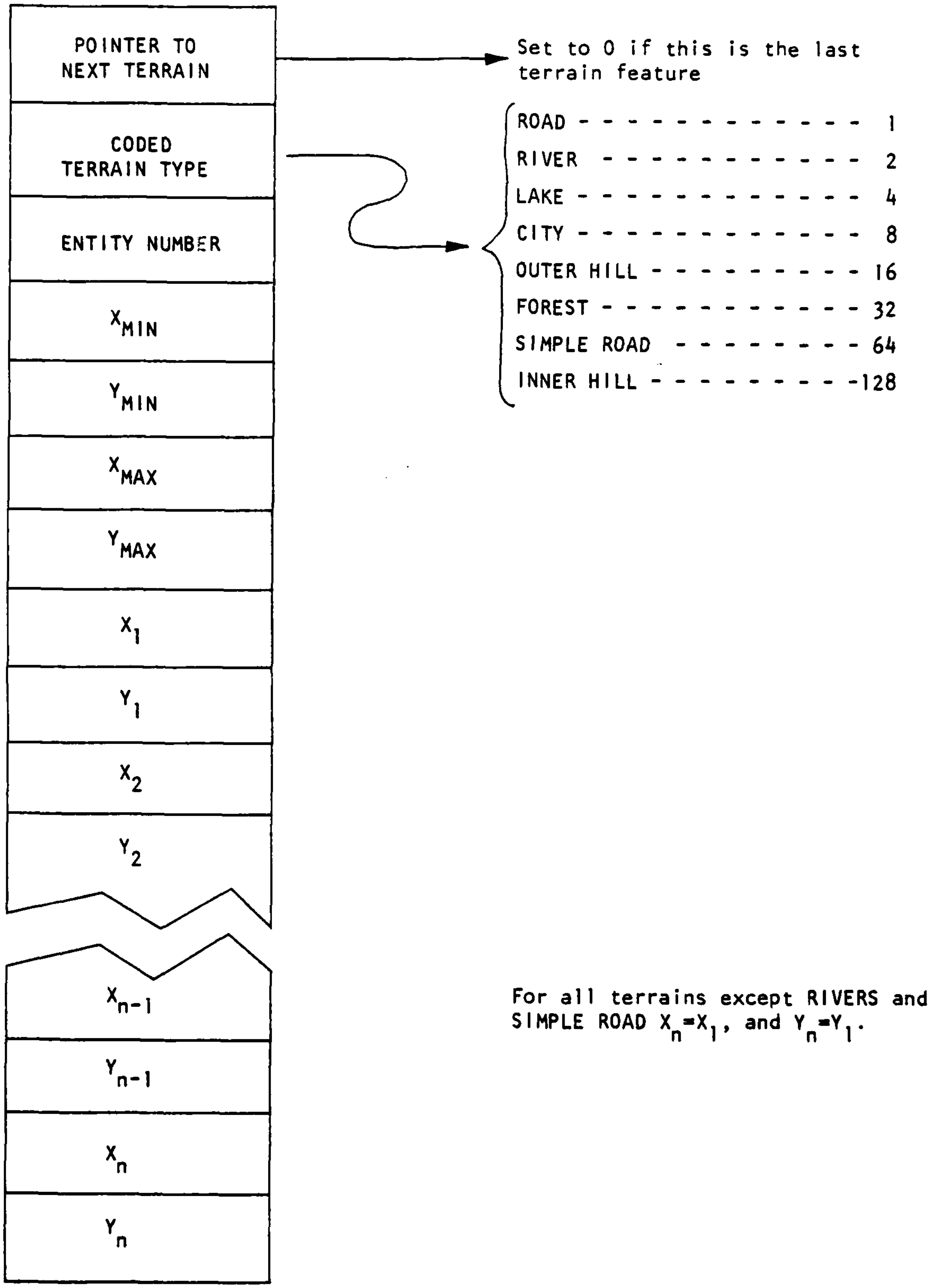


Figure 3-2. Terrain Information Organization.

words which detail (1) the number of green units, (2) the number of red units, (3) a pointer to the first green unit, and (4) a pointer to the first red unit. Each unit is allotted an area in the MDF of length RCDSIZ (currently 20). The organization of these unit information records is shown in Figure 3-3.

Following this unit status information is a sequence of unit action blocks, one block per unit. As seen in Figure 3-3, the last word of each unit information record is a pointer to that unit's action block. The action block for a given unit contains a header followed by a time-ordered sequence of action records which specify the unit's movement through arrival times at nodes and terrain transitions. The action records are variable in length to allow inclusion of other time-dependent actions such as changes in mission and supply level. The type of action and length of the record are specified by a bit-code or "vector change status word" at the beginning of each record. This is shown in detail in Figure 3-4. Note that detection and engagement events were to be stored here in the initial design. Since then, they have been moved to a separate area in the MDF to make for easier editing of these events. Also note that the pointers in the unit action block header are relative to the beginning of the action block to allow for easier editing of unit movements.

The event file begins at MDF (MDF(4)) with a three word header. The first word is a pointer to either the next event to be recalled (when in master replay) or the first word of a new event to be stored. The second word of the header is the time (in minutes) up to which events have been determined. The third word is not used at the present time. Events take up five words of the MDF and are stored in chronological order (See Figure 3-5). The first word gives the time of the event, the second word gives the event code (1=friendly detects, 2=enemy detects, -1=friendly no longer detects, -2=enemy no longer detects). The third word identifies the friendly unit by its order in the MDF, and the fourth word indicates the red units in bit-coded form. The last word in the event file is at MDF (MDF(3)-1). At MDF (MDF(3)) the scratch area begins.

0	LABEL
1	ENTITY
2	SIZE
3	TYPE
4	X POSITION
5	Y POSITION
6	TERRAIN
7	MOVING/STATIONARY
8	ASSETS REMAINING
9	ENDURANCE
10	DETECTED
11	UNITS
12	UNITS IN
13	FIRING RANGE
14	ENGAGED
15	UNITS
16	DETECTING
17	UNITS
18	MISSION/STATUS
19	POINTER TO ACTION RECORD

Label: 0 - 30

Entity #: 200 - 299

Size: 1 Brigade
2 Battalion
3 Company/Battery

Type: 1 Artillery
2 Armor
3 Infantry
4 Mechanized Infantry

Terrain Type: Bit coded See Figure 3-2

Moving/Stationary (1,0)

Detected Units: Set of opposing units in Bit Coded form

Units in Firing Range: "

Engaged Units: "

Detecting Units: "

[Mission/Status

<u>Active Unit</u> (0 - 1)
-10 <u>Defense</u>
- 8 <u>Defense</u>

<u>Inactive Unit</u>
0 → 1440 Scenario Time of Extension
-1 Indicates unit is offscreen-action entries will at some scenario time enter the unit

Figure 3-3. Unit Information Record.

	0	Ptr. to Last Action	← unit information record pts. here
Header	1	Ptr. to Present Action	
	2	Ptr. to Next Action	
	3	Vector change status word	Initial Unit State (at problem time zero)
	4	X position	
	5	Y position	
	6	time of arrival/transition	
	7	terrain type	
	8	detected units 1-15	
	9	detected units 16-30	
	10	units in firing range 1-15	
	11	units in firing range 16-30	
	12	engaged units 1-15	
	13	engaged units 16-30	
	14	resupply level (%)	
	15	mission	

Unit Action Record

0 vector change status word*
1 X position
2 Y position
3 time of arrival/transition
4 terrain type (always included)
• detected units 1-15
• detected units 16-30
• units in firing range 1-15
• units in firing range 16-30
• engaged units 1-15
• engaged units 16-30
• resupply level (%)
• mission

— relative position in list if all words contained a change.
*Only those words which indicate a transition will be included in any given vector.

Vector Change Status Word

|0|0|n|m|l|k|j|i|h|g|f|e|d|c|b|a|

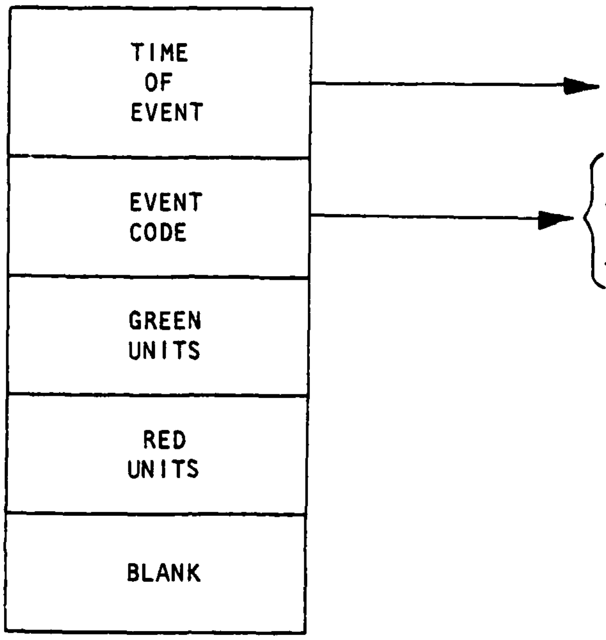
If bit is set it denotes that:

a unit will arrive at recorded X,Y at the recorded arrival time (always set)
b unit will arrive at recorded terrain type at the recorded arrival time (always set)
c there is a newly detected enemy unit, labeled 1-15, at the record transition time
d there is a newly detected enemy unit, labeled 16-30, at the record transition time
e there is an enemy unit within firing range, labeled 1-15, discovered at the recorded transition time
f there is an enemy unit within firing range, labeled 16-30, discovered at the recorded time
g an engagement has commenced with an enemy unit, labeled 1-15, at the recorded time
h an engagement has commenced with an enemy unit, labeled 16-30, at the recorded time
i this unit has been resupplied at the recorded transition time
j this unit has a new mission definition at the recorded transition time
k unit is active and able to participate in the scenario
l always set
m,n set as needed for proper bit count (bit positions total number of words in action record vector)

If bit is not set it denotes that:

a unit X,Y destination in this word is identical to previous vector; unit will wait at this position until the recorded transition time
b,c,d,e,f,g,h,i,j there is no change in the unit state for the corresponding descriptor words from the previous vector - (there are no words containing description of these states in this vector)
k unit does not participate in any scenario activities; it is either offscreen or expended in battle

Figure 3-4. Unit Action Block.



In Minutes

1: Green Detect Reds
-1: Green No Longer Detects Reds
2: Reds Detect Green
-2: Reds No Longer Detect Green

Unit number refers to location in the MDF to indicate multiple Red Units bit coding is used. For example, to denote units 1, 3, and 5 the code would be

$$2^{(1-1)}+2^{(3-1)}+2^{(5-1)} = 21$$

Figure 3-5. Event Record.

3.1.2 Display File

All information to be displayed to the user must be stored in the 4900-word display file. The display file is actually a program which is executed by a separate processing unit contained in the display generator. The commands in the display file are divided into subfiles called "entities," which are identified by a unique entity number. In general, logically distinct elements of the display are separate entities in the TOMM display file. For example, each terrain feature is a separate entity, as is each unit in the order of battle. Figure 3-6 is a list of the entities in the TOMM display file, with their entity numbers and description. The chief advantage to the organization of the display file into entities is that, once created, entities may easily be made to move on the screen, turned off and on, blinked, changed in color, have different alphanumeric messages displayed, and so on. This flexibility is used extensively in TOMM.

3.1.3 Other Data Structures

Although the Master Data File and the display file are the most important data structures in TOMM, there are several others that will be briefly explored here. A five-word array called OVLYKY (in common block OVRLAY) is used to record overlay sequencing information. The number of the next overlay of TOMM to be called is stored in OVLYKY(1). If several overlays are needed to perform a desired function, their numbers are stored in sequence beginning with OVLYKY(1) up to a limit of five. This method must be used since an overlay cannot call another overlay directly. All overlay sequencing must be done by the root segment of TOMM, which obtains the necessary sequence from OVLYKY.

Another array called IATT (of length twelve words) is used by the FORTRAN graphics package to record information about display interrupts. For example, pressing or releasing one of the function keys causes a hardware interrupt to be generated which halts the display and causes information about the type of interrupt and key number to be stored in IATT.

Entity Number(s)	Description
1	Green cursor
2	Scenario border
3	Time-of-day scale
4	Time-of-day numerals and pointer
5	Unit trackball ID circle
8- 17	Cluster ID circles
100-199	Terrain features
200-299	Units
300-399	Reference lines
400-410	Future position contours
450-499	Future position units
1000-1010	Left-side-of-screen character areas (menu area)
1011-1040	Relocatable character areas
1300-1399	AOBE's
1700-1799	Detection contours
1900-1999	Unit-to-unit detection lines (current replay)
2100-2199	Unit-to-unit detection lines (master replay)
2301-2302	Unit mobility illustrations
3000-3999	Movement paths and arrival-time character areas

Figure 3-6. TOMM Display File Entities.

Finally, there is a disk file called SCENES which is used to store scenarios 1 through 4. Each scenario is stored as a copy of the Master Data File (MDF), together with the current problem time and the next unused unit entity number. This is sufficient information for TOMM to recall the scenarios at any later date.

3.2 DATA FLOW

Figure 3-7 shows the high-level flow of data in TOMM. All user entry of data occurs through the function keyboard and the trackball next to the display (with the exception of the deletion of reference lines, which uses the lightpen). The function keyboard is used for the selection of the next function to be executed, for the selection of an item from a screen menu, and for numerical data entry. The trackball is used to input screen coordinates for the drawing of terrain, the placement and selection of units, and the definition of unit movement paths.

With the principal exception of control information (selection of function), the user data input as above is passed to the Master Data File (MDF). This takes place, for example, in terrain definition, order-of-battle definition, movement definition, and changes therein such as adding or deleting units or movements. Terrain and order-of-battle definition are also stored in the display file. Detection information is generated automatically during current replay mode and passed to the MDF. The MDF itself may be stored in the disk file SCENES, or another MDF may be recalled from the disk file.

Displayable information passes from the MDF to the display file when not directly input by the user. This occurs during the replay of unit movements (and also the replay of detections in master replay mode), and when an old MDF is recalled from the SCENES file for display. Information is never passed from the display file to the MDF.

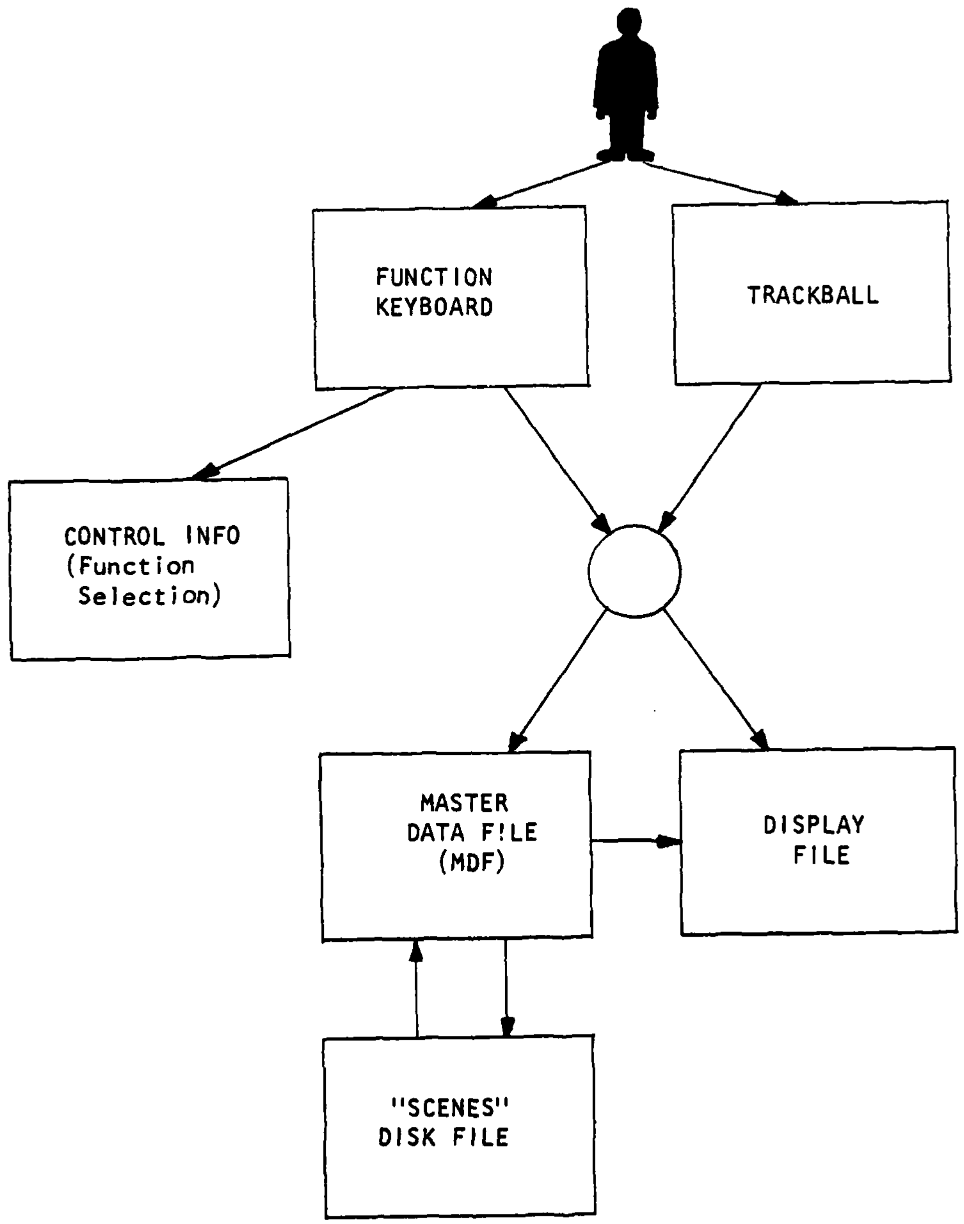


Figure 3-7. High-Level Flow of Data.

4.0 MAJOR SOFTWARE MODULES

This section contains write-ups on the major subprograms in TOMM. Criteria for inclusion were importance, length, and complexity of the subprogram. As a result, 26 routines were chosen for detailed description in the following pages, arranged in alphabetical order. Each write-up includes the name of the subprogram at the top of the page, followed by:

(1) a list of the overlays in which it is used
(2) a description of the purpose of the subprogram
(3) a description of the method used
(4) a list of the subprograms called (not including system routines such as the FORTRAN graphics package)
(5) the calling sequence
(6) a description of the subprogram arguments, if any.

ADDREF

USED IN OVERLAY(S): 3

PURPOSE: To draw connected line segments, reference lines, on a scenario where they remain until they are either explicitly erased or a new scenario is recalled or defined.

METHOD: REFENT is a variable in COMMON initialized to 300 which indicates the entity number of the next (or first) reference line. The cursor is turned on and is relocated by the subroutine MOVENT when the trackball is moved. The accept key is used to set a node and the return key is hit to end a given reference line. REFENT is incremented by 1 for each reference line created.

SUBROUTINES CALLED: LAMPS MOVENT CKINT

CALLING SEQUENCE: CALL ADDREF

ARGUMENTS: None.

ADDUNT

USED IN OVERLAY(S): 7

PURPOSE: Allow user to add units to scenario.

METHOD: User selects unit type, size, and side from menus. A unit then appears on the screen which is placed with trackball and accept/reject keys. If the unit is accepted, a unit information record and a unit action block are created for it. Each in turn is inserted in the MDF using subroutine MDFSHF to shift the MDF to make room for the new unit. The unit count and all MDF pointers are also updated (the latter in MDFSHF). The procedure for adding a new unit at other than problem time zero is incompletely integrated with the rest of TOMM and should not be used.

SUBROUTINES CALLED: USTMSG, SELUTT, SELUTS, SELSID, COLOR, LAMPS, ADUMSG, ONSEL, MOVENT, CKINT, OFFSEL, BLKSEL, IWRAMI, MDFSHF

CALLING SEQUENCE: CALL ADDUNT.

ARGUMENTS: None.

AOBE

USED IN OVERLAY(S): 13

PURPOSE: Determine and plot the closed contour which represents the area of battlefield effectiveness for a given unit. Blink both the chosen unit and its AOBE. Erase the contour from the display file when return key is hit.

METHOD: Seventy-two line segments are created originating at the unit location, set 5° apart, with length equal to the maximal fire power distance. Each segment is truncated at the "side" of a hill (see HILSDS) if a crossing occurs. The subroutine TTLEGS determines the terrain which this line segment crosses. This information is passed to NDAOBE which returns (IX2, IY2), the furthest point of effective fire along the given line segment. This results in seventy-two points which are connected to create the contour.

SUBROUTINES CALLED:	WHEREX	NDAOBE	GSAVE	IWRAMI
	DELAY	OFFSEL	TTLEGS	GREST
	BLKSEL			

CALLING SEQUENCE: CALL AOBE (UNTPTR)

ARGUMENTS: UNTPTR: The absolute integer address in the MDF of the chosen unit.

CLSARL

USED IN OVERLAY(S): 12

PURPOSE: Computes and displays minimum arrival times at path nodes for cluster movement, allows user to modify these times, and transfers corresponding action records into each cluster member's action block in the MDF.

METHOD: The method used is identical to that of UNTARL in overlay 11, except that the maximum speed on each leg is defined to be the slowest maximum speed for any cluster member during any terrain component of that cluster member's movements during that leg. Note that a cluster member may execute several changes of direction during a single path leg as shown on the screen (especially if the cluster member is a following unit in column movement). Subroutine CTMTBL performs the actual minimum arrival time computation.

SUBROUTINES CALLED: PNTDST CTMTBL TTABLE INSERT

CALLING SEQUENCE: CALL CLSARL

ARGUMENTS: None

CONGEN

USED IN OVERLAY(S): 16

PURPOSE: To generate and draw future position contours using the minimum-arrival-time grid stored on disk in file VMATRX by overlay 15.

METHOD: The contours are generated using ACM Algorithm 531, i.e., subroutine GCONTR. See the ACM documentation for details of the method used.

SUBROUTINES CALLED: GCONTR

CALLING SEQUENCE: CALL CONGEN

ARGUMENTS: None

DEFTER

USED IN OVERLAY(S): 4

PURPOSE: Define terrain features and store them in the MDF and the display file.

METHOD: Clears MDF, resets pointers, and inputs terrain using menu selection of terrain type followed by input using trackball and accept key. User may then accept or reject the current feature. If rejected, the display entity is turned off, but no other changes or deletions are made. Roads drawn are converted to corridors using subroutine ROAD. On exit, sets MDF(2) to end of terrain plus one .

SUBROUTINES CALLED: SELTER, TCOLOR, DFTRMS, TERRNID, ONSEL, SETPNT, GDRAW, ROAD, BLKSEL, LAMPS, CKINT, OFFSEL

CALLING SEQUENCE: CALL DEFTER

ARGUMENTS: None.

DELUNT

USED IN OVERLAY(S): 7

PURPOSE: Allows the user to delete units at problem time zero.

METHOD: The unit to be deleted is selected with the trackball circle, trackball, and accept key. The unit action block and unit information record for the selected unit are deleted in turn from the MDF using subroutine MDFSHF to shift the MDF. The unit entity is turned off, the unit count is corrected, and the MDF pointers are updated (in MDFSHF).

SUBROUTINES CALLED: USTMSG, ONSEL, SELUNT, MDFSHF, OFFSEL, BLKSEL

CALLING SEQUENCE: CALL DELUNT.

ARGUMENTS: None.

DETECT

USED IN OVERLAYS: 19

PURPOSE: This logical function is set to true if a unit, located at IX1,IY1 with a presence radius of RADII(1), has a line of sight which extends unobstructed to a point RADII(2) units from the point IX2, IY2. RADII(2) will be set to the detectability radius of a unit and this function, in effect, determines if the former unit detected the latter.

METHOD: The method used is virtually identical to the algorithm used in ENDLOS except that DETECT returns either true or false.

SUBROUTINES CALLED: NONE

CALLING SEQUENCE: DETECT(IX1,IY1,IX2,IARY,RADII)

ARGUMENTS:

IX1,IY1:	location of first unit
IX2,IY2:	location of second unit
IARY:	array with data concerning terrain intersections of the line segment (see TTLEGS)
RADII(1):	first unit's presence radius
RADII(2):	second unit's detectability radius

DRECTR

USED IN OVERLAY(S): 1

PURPOSE: This subroutine controls almost all sequencing of overlays and acts as an interface between overlays. Upon completion of a major function, control returns to DRECTR and remains there until the user selects a new option.

METHOD: The manipulation of the integer array OVLYKY is the principle framework behind DRECTR. OVLYKY is used as a stack to order the calling of overlays. Each time an overlay is exited, then the root calls the overlay referenced by OVLYKY(1). An example of overlay sequencing occurs when KEY=DFTER (the user wishes to define terrain) in DRECTR and OVLYKY(1)=4, OVLYKY(2)=5, OVLYKY(3) = 6.

	OVLYKY(1)	OVLYKY(2)	OVLYKY(3)
	4	5	6
ROOT calls Overlay 4 which is executed and sets OVLYKY(1) = 1	1	5	6
ROOT calls Overlay 1 (DRECTR) which pops the stack	5	6	0
ROOT calls overlay 5 which is executed and sets OVLYKY(1)=1	1	6	0
ROOT calls overlay 1 (DRECTR) which pops the stack	6	0	0
ROOT calls overlay 6 which sets OVLYKY(1)=1	1	0	0
ROOT returns to DRECTR which waits in a loop until a new option is chosen	1	0	0

DRECTR continued

SUBROUTINES CALLED: SCINIT LAMPS SELFIL GSAVE

CALLING SEQUENCE: CALL DRECTR

ARGUMENTS: NONE

DTCNTR

USED IN OVERLAY(S): 18

PURPOSE: Determine and plot the closed contour which represents the area a chosen unit can see. Blinks both the unit and the closed contour until the return key is pushed, then erases the contour from the display file.

METHOD: Seventy-two line segments are created originating at the unit location, set 5° apart, with length equal to the maximal line of sight. Each segment is truncated at the "side" of a hill (see HILSDS) if a crossing occurs. The subroutine TTLEGS determines the terrain which this line segment crosses. This information is passed to ENDLOS which returns IX2, IY2, which is the end of the line of sight along the given line segment. This results in seventy-two points which are connected to create the contour.

SUBROUTINES CALLED:				
	WHEREX	OFFSEL	GSAVE	IWRAMI
	BLKSEL	ENDLOS	TTLEGS	GREST

CALLING SEQUENCE: CALL DTCNTR (UNTPTR)

ARGUMENTS: UNTPTR: The absolute integer address in the MDF of the chosen unit.

ENDLOS

USED IN OVERLAY(S): 18

PURPOSE: To determine how far a unit's line of sight extends from IX1,IY1 to IX2,IY2 given the terrain crossings in the array IARY. Return the last visible point in the parameters IX2,IY2.

METHOD: If the center of the unit is on a hill and IX1,IY1 is on clear terrain, then the routine proceeds as though IX1,IY1 were located on a hill. The variable IHILL is used to keep track of the number of times an outer hill is crossed. The subroutine jumps through the transition matrix LOSMTX until either a dense terrain is encountered at X_N, Y_N or it is determined that the line of sight extends a distance of LEN1. In either case, IX2, IY2 are set and ENDLOS is exited.

SUBROUTINES CALLED: NONE

CALLING SEQUENCE: CALL ENDLOS(IX1,IY1,IX2,IY2,IARY)

ARGUMENTS:

IX1,IY1: the intersection point of a unit's presence circle and the line extending to the point IX2,IY2.

IX2,IY2: on input, the end of the line segment; on output, the last visible point along this line segment.

IARY: an array which details the terrain crossings along the line segment (see TTLEGS).

FORCES

USED IN OVERLAY(S): 1, 2

PURPOSE: This routine is called to create a picture of a unit of given size and type.

METHOD: Before FORCES is called the entity which will become the unit must either be referenced (with GENT) or initialized (with GBEG). Depending on the size and type the unit is drawn. The units are of length 32 raster units and height 16 raster units.

SUBROUTINES CALLED: None.

CALLING SEQUENCE: CALL FORCES (ISIZE, ITYPE, IEL).

ARGUMENTS:

ISIZE: The size of the unit
- 1 Brigade
- 2 Battalion
- 3 Company

ITYPE: The type of the unit
- 1 Artillery
- 2 Armor
- 3 Infantry
- 4 Mech. Infantry

IEL: The next available element number (for both input and output).

HILSDS

USED IN OVERLAY(S): 17, 19

PURPOSE: Each hill has two "sides" which will obstruct both a unit's line of sight and firing effectiveness. These "sides" are different for every unit, depending on the unit's location. This subroutine determines these sides for any given unit.

METHOD: For each hill, the function IOCT is called. This will return an integer indicating the relative position of the inner hill with the unit involved (see chart below). For simplification, if IOCT returns 0, the given hill is ignored. Otherwise, a search begins for the nodes of the inner hill having slopes (with respect to the given unit) with the qualities listed in the chart (e.g., when IOCT = 5, the nodes with the greatest and least slopes are found). The two "hill sides" will originate at these two nodes, will be perpendicular to the line from the unit to the node, and will terminate at the outer hill. For each hill, the two sides are stored as four points (8 integers) in the integer array SIDES. SIDES(1) is set to the number of pairs of "sides."

	IOCT = 9 greatest and least slope	IOCT = 8 least positive and greatest negative slope	IOCT = 10 greatest and least slope
y max	IOCT = 1 greatest and least slope	IOCT = 0 hill ignored	IOCT = 2 greatest and least slope
y min	IOCT = 5 greatest and least slope	IOCT = 4 least positive and greatest negative slope	IOCT = 6 greatest and least slope
		x min	x max

SUBROUTINES CALLED: IOCT TTLEGS

CALLING SEQUENCE: CALL HILSDS (UNTPTR)

ARGUMENTS: UNTPTR: Integer pointer to the absolute address in the MDF of the given unit.

MVMMI

USED IN OVERLAY(S): 8

PURPOSE: To perform the initial stages of movement definition: selection of type of movement, unit(s) involved, any cluster centers or reference points necessary to the movement type, and finally drawing of the movement path itself.

METHOD: A set of temporary entities are created for movement path and arrival time display. The user selects the movement type from a menu, specifies the unit(s) involved using the trackball circle, trackball, and accept key, and specifies any cluster centers or reference points using a cursor and the trackball. Finally, the user inputs the movement path using the trackball and accept key in subroutine DRWPTH. A maximum of ten legs are allowed, although less may be drawn by hitting the return key when done. The path nodes are stored temporarily in ARRAY in blank common, where ARRAY(1) contains the number of legs. Column 1 of array UNTPTR (common CLUSTR) is used to store pointers to the MDF unit information records of cluster members if cluster movement is involved.

SUBROUTINES CALLED: COLOR, SELMVT, GREST, SELUNT, IDUNTS, CLCMSG, ONSEL, SETPNT, OFFSEL, BLKSEL, RFPTMG, DRWPTH

CALLING SEQUENCE: CALL MVMMI.

ARGUMENTS: None.

NDAOBE

USED IN OVERLAY(S): 13

PURPOSE: Given line segment (IX1, IY1), (IX2, IY2) and UTYPE (1 = artillery, 2 = armor, 3 = infantry, 4 = mechanized infantry), determine the point along this line which is the end of effective fire power.

METHOD: A unit of type UTYPE positioned in terrain CT firing into terrain NT has an effective firing range of AOBEMX (CT, NT, UTYPE). If this distance would extend the unit's firing range beyond the outside border of terrain NT into terrain NT1, then the firing range is the maximum of the terrain NT border or AOBEMX (NT, NT1, UTYPE). If this distance also exceeds the distance to the outside border of NT1, then the matrix AOBEMX is consulted again for a new distance. This process continues until the end of effective fire power is found.

SUBROUTINES CALLED: TTYP

CALLING SEQUENCE: CALL NDAOBE (IX1, IY1, IX2, IY2, UTYPE, IARY)

ARGUMENTS: (IX1, IY1) → (IX2, IY2): The line segment in question.

UTYPE: Unit type -- 1 for Artillery 2 for Armor
3 for Infantry 4 for Mechanized Infantry

IARY: An array detailing intersections:

IARY(1) = # of intersections
IARY(3) = terrain type of IX1, IY1
IARY(5) = X_1 (1st intersection)
IARY(6) = Y_1 (1st intersection)
IARY(8) = new terrain type
IARY(5N) = X_N (Nth intersection)
IARY(5N+1) = Y_N (Nth intersection)
IARY(5N+3) = new terrain type

REHASH

USED IN OVERLAY(S): 21

PURPOSE: Master replay calls this routine at every scenario time that an event has occurred. REHASH updates the unit information records, draws lines between detecting and detected units, and blinks detected units. REHASH is exited after all events occurring at the given scenario time are processed.

METHOD: There are four sections of REHASH which deal with the four different types of events stored in the event file: green detects reds (event 1), green no longer detects reds (event -1), green is detected by reds (event 2), green no longer is detected by reds (event -2). The unit information records are changed, and if the event is a detection, a line (of appropriate color) is drawn between the units. When the last event has been processed, BLKUTS is called to blink (or stop blinking) the units. There is a pause to facilitate the viewing of the detection lines and control returns to REPLAY after these lines have been erased from the display file.

SUBROUTINES CALLED: GSAVE DELAY

BLKUTS GREST

CALLING SEQUENCE: CALL REHASH

ARGUMENTS: NONE

REPLAY

USED IN OVERLAY(S): 21

PURPOSE: When this routine is called for current replay, units are moved according to their previously defined paths, and new detections and end of detections are stored in both the event file and the respective unit information records. When used for master replay, the only difference from the above is that events are recalled from the event file rather than determined by algorithms.

METHOD: The time clock and movements are updated by the routines TMBASE (which has a delay determined by replay speed), MVMENT, DSPLAY and CLKUPD. When master replay is running, events are recalled by REHASH, and when current replay is used, overlay 19 is called every 10 "minutes" of scenario time to determine detections.

SUBROUTINES CALLED:				
	MESAGE	CKINT	TMBASE	DSPLAY
	DCTOUT	RPLSPD	RESULT	CLKUPD
	BLKUTS	REHASH	MVMENT	DINTRP

CALLING SEQUENCE: CALL REPLAY

ARGUMENTS: NONE

RISE

USED IN OVERLAY(S): 15

PURPOSE: Compute a minimum-arrival-time grid V for one or more units from the terrain information stored in the MDF. (V is actually a 27 X 27 array containing a 25 X 25 grid with a "border"). For a single unit, each element of V contains the minimum time required for that unit to arrive at the corresponding grid point on the screen, given the unit's initial position as specified by the operator. For multiple units, each element of V contains the smallest minimum arrival time for any of the units. This matrix V is stored on disk to be used by overlay 16 to compute future postion contours (see CONGEN).

METHOD: The method used is the "rising water" algorithm described in detail in Appendix C of ISC Report 271-1. Two main changes have been made: (1) the matrix used to store the actual minimum-time path directions has been deleted since it is not needed, and (2) in order to handle multiple initial "surf" points (i.e., multiple units), each element of V is temporarily incremented by one and multiplied by 2^{30} before processing the next unit.

SUBROUTINES CALLED:	RIVRD	GETG	GAMMOD
	NTABLE	SRFAD	FOM
	DLTSRF	SCHST	OUT2

CALLING SEQUENCE: CALL RISE (DELX, DELY)

ARGUMENTS: DELX = horizontal grid spacing (distance between neighboring grid points) in raster units.

DELY = vertical grid spacing in raster units.

ROAD

USED IN OVERLAY(S): 4

PURPOSE: Converts simple road paths as input by the user into closed corridors. The simple road becomes terrain type 7. The new closed-corridor road is stored in the MDF as terrain type 1 immediately following the simple road.

METHOD: Given a line segment from (X_1,Y_1) to (X_2,Y_2),

let $X = -(ad+bc')/r^2$

$Y = (ac'-bd)/r^2$

where $a = Y_1-Y_2$

$b = X_2-X_1$

$c = X_1Y_2-X_2Y_1$

$r = \sqrt{a^2+b^2}$

$d = c\pm5r$

then (X,Y) defines a point 5 units from (X_1,Y_1) along a perpendicular (to the right if $d = c-5r$, to the left if $d = c+54$). Now given a third point (X_3,Y_3), let $X = (b'd-df')/\Delta$

$Y = (ad'-a'd)/\Delta$

where $a' = Y_2-Y_3$

$b' = X_3-X_2$

$c' = X_2Y_3-X_3Y_2$

$r' = \sqrt{a'^2+b'^2}$

$d' = c'\pm5r'$

$\Delta = a'b-ab'$

Then (X,Y) defines a point to one side of (X_2,Y_2) which is the intersection of lines parallel to $(X_1,Y_1) \Rightarrow (Y_2,Y_2)$ and $(X_2,Y_2) \Rightarrow (X_3,Y_3)$ and 5 units away. Special handling is used for $\Delta<1$.

ROAD (Continued)

SUBROUTINES CALLED:

CALLING SEQUENCE: CALL ROAD(TPTR)

ARGUMENTS: TPTR = pointer to start of simple road in MDF.

SCINIT

USED IN OVERLAY(S): 1, 2

PURPOSE: SCINIT clears the screen and the display file, and sets up all permanent entities for a scenario.

METHOD: A call to GINI clears both the screen and the display file. The cursor, border, movie time thermometer, and time pointer are drawn. A unit of each size and type is created but is not shown on the screen. When new units are created these prototypes are copied. Character display areas are also set up.

SUBROUTINES CALLED:

GINI	RPLSPD	MBTBL
SHOCUR	FORCES	

CALLING SEQUENCE: CALL SCINIT

ARGUMENTS: None.

SETIME

USED IN OVERLAY(S): 20

PURPOSE: This subroutine sets the scenario time. Units are relocated, blinked if they are being detected, and have their unit information records updated.

METHOD: Time is first set to zero. URESET relocates the units and resets the unit information records. The scenario is replayed minute by minute as MVMENT and DSPLAY update unit positions, CLKUPD advances the clock, UPDATE checks the event file to keep the unit information records current, and BLKUTS blinks detected units.

SUBROUTINES CALLED:	SELTIM	DISPLAY	BLKUTS	RESULT	OFFSEL
	URESET	CLKUPD	UPDATE	MVMENT	BLKSEL

CALLING SEQUENCE: CALL SETIME

ARGUMENTS: NONE

TERPTH

USED IN OVERLAY(S): 10

PURPOSE: Takes individual unit paths stored in PARRAY (common SCRTCH), inserts any terrain crossings, and stores the resulting sequence of path nodes in the MDF as unfinished 5-word records corresponding to unit action records.

METHOD: Path nodes are read from PARRAY and calls to subroutine TTLEGS are made to find terrain crossings and store the results as five-word records in the MDF. (Note that path nodes for non-lead-unit column movements were stored three words apiece in PARRAY by overlay 9). One word of each five-word record contains the corresponding leg number for the path drawn on the screen (for non-lead-unit column movement this information was stored as the third word of each path node; for all other movements this information can be computed by TERPTH). Column 2 of array UNTPTR is used to store pointers to MDF path information created by TERPTH.

SUBROUTINES CALLED: TTLEGS

CALLING SEQUENCE: CALL TERPTH

ARGUMENTS: NONE

TTLEGS

USED IN OVERLAY(S): 5, 10, 17, 18, 19

PURPOSE: Given the end points of a line segment, determine all terrain crossings. Only those terrains indicated by JLOOK (in bit coded form) are examined.

METHOD: Check MDF for appropriate terrains. If the line segment does not pass through the min-max square of terrain, ignore this terrain. Otherwise, check for any intersection with each line segment by calling WHEREX. Store each intersection in IARY. When all terrain has been processed, IARY is reorganized so that the list of intersections is in order.

SUBROUTINES CALLED: WHEREX SWPSUM

CALLING SEQUENCE: Call TTLEGS (X1, Y1, X2, Y2, ITYPE, JLOOK, IARY, LEN)

ARGUMENTS: X1, Y1, X2, Y2: line segment

ITYPE - Must be set by calling routine to the terrain Type of X1, Y1. If so set, IARY (3) returns this terrain type masked by JLOOK.

JLOOK - Bit mask used to indicate which terrain types are to be examined or ignored when looking for crossings Set bits 1 (LSB) to 6 (1 = Road, ..., 6 = Forest).

IARY - An array in which crossings can be passed back to the calling program.

LEN - Length of IARY (number of words)

IARY(1) = Number of intersections found
= Negative of number of points found (if LEN is too small for all of the points to be found)
= -1000 (if LEN is too small for any points to be found.)

IARY(3) - Terrain type of first point in bit coded form masked by JLOOK.

TTLEGS (Continued)

IARY(5) = X of first point of intersection.

IARY(6) = Y of first point of intersection.

IARY(7) = Address of terrain in the MDF.

IARY(8) = Terrain type of first point of intersection in bit coded form.

A cell is 5 words long starting at IÁRY(4), et cetera.

UNITS

USED IN OVERLAY(S): 6

PURPOSE: Set up initial order of battle following terrain definition. Friendly and then enemy orders of battle are established and stored in the MDF and the display file.

METHOD: The user selects unit type and then size, from a menu, followed by placement with trackball and accept or reject key. Rejected units are turned off and are effectively erased from the MDF. For each accepted unit, a unit information record is added to the MDF. After all units have been defined, action blocks for each unit are added to the MDF. Then MDF(3) is set to the end of the action blocks plus one.

SUBROUTINES CALLED: OOBMSG, SELUTT, SELUTS, COLOR, LAMPS, ADUMSG, ONSEL, MOVENT, CKINT, OFFSEL, BLKSEL, IWRAMI, ACTBLK

CALLING SEQUENCE: CALL UNITS

ARGUMENTS: None.

UNTARL

USED IN OVERLAY(S): 11

PURPOSE: Computes and displays minimum arrival times at path nodes for single-unit movement, allows user to modify these times, and transfers corresponding action records into the unit's action block in the MDF.

METHOD: Each path node is represented by an unfinished five-word record created by overlay 10. Included in each record is a word indicating on which leg of the displayed path the given node falls (each path leg may have multiple nodes on it due to terrain crossings). For each display path leg, the maximum speed on each (terrain) subleg is computed, leading to a minimum elapsed time for the leg and hence minimum arrival times for the displayed path nodes. Subroutine TTABLE allows the user to modify the overall path speed or individual arrival times. Finally, subroutine MDFSHF creates the appropriate space in the unit action block and the now completed five-word action records are transferred to this space. All MDF pointers are updated.

SUBROUTINES CALLED:	PNTDST	RUNTSP	TTABLE
	MDFSHF	MVDONE	BADPTH

CALLING SEQUENCE· CALL UNTARL

ARGUMENTS: None

UNTMOB

USED IN OVERLAY(S): 5

PURPOSE: Create the unit terrain mobility illustrations (cross-hatching) in the display file and turn them off until requested later.

METHOD: Parallel lines are created and examined in turn using TTLEGS to find terrain transitions and store them temporarily in the MDF scratch area. Subroutine DRTMOB is used to convert the terrain transitions into the appropriate line structure changes and store the result in the display file. Note that the accompanying alphanumeric table is not generated here, but instead by MBTBL during scenario initialization (since the table entries are independent of the scenarios).

SUBROUTINES CALLED: COLOR, IWRAMI, TTLEGS, DRTMOB

CALLING SEQUENCE: CALL UNTMOB

ARGUMENTS: None.

UNTPTH

USED IN OVERLAY(S): 9

PURPOSE: Convert movement paths stored in ARRAY (blank common) into individual unit paths stored in PARRAY (common SCRTCH).

METHOD: No conversion is needed for single unit movement, only simple transfer from ARRAY to PARRAY. For each of the 4 types of cluster movement, a different subroutine performs the conversion of the movement path as drawn on the screen into the actual movements performed by an individual unit in the cluster. Cluster units are located using the pointers in column 1 of array UNTPTR created by MVMMI in overlay 8. Column movement units other than the lead unit have each path node labeled with an extra word indicating the corresponding lead unit leg being executed.

SUBROUTINES CALLED: SNGUNT, DIRDEP, FIXDEP, COLUMN, LPFROG

CALLING SEQUENCE: CALL UNTPTH

ARGUMENTS: None.

UUDCT

USED IN OVERLAY(S): 19

PURPOSE: This subroutine is called during current replay to check all pairs of green and red units to determine if either a new detection has occurred or if a unit is no longer being detected by an opposing unit. The unit information records and the event files are updated. If no new detections have occurred, REPLAY again is called; if detections have occurred, appropriate lines are drawn between the units and control remains in a loop at the end of UUDCT until the accept or reject button is pushed. Then the lines are erased from the display file and control passes to REPLAY once again.

METHOD: UUDCT first checks detections between green Unit 1 and all the opposing units, then checks green Unit 2 with all opposing units, and so on. HILSDS, which stores line segments representing "hill sides" in the array SIDES, is called for each green unit. When a "hill side" lies between two units, line of sight is obstructed. If no "hill side" exists between the two units, then TTLEGS is called, which returns in the array IARY the terrain crossings of a line between the two units. This information is sent to the logical function DETECT which uses a transition matrix to determine if the green unit can see the red unit. UUDCT then alters IARY so that the line originates at the red unit and terminates at the green unit. DETECT is called once again to check if the red unit detects the green unit.

SUBROUTINES CALLED: HILSDS BLKUTS WHEREX

GSAVE DETECT GREST

CALLING SEQUENCE: CALL UUDCT

ARGUMENTS: NONE

WHEREX

USED IN OVERLAY(S): 5, 10, 17, 18, 19

PURPOSE: Given two line segments, determine if and where they cross.

METHOD: Reject those cases where one line segment lies entirely to the left or the right or entirely above or below the other. There is a separate section for each of the four cases: (1) both lines vertical, (2) one line vertical and the other with a defined slope, (3) both lines having the same slope, and (4) both lines having different slope.

SUBROUTINES CALLED: None.

CALLING SEQUENCE: CALL WHEREX (IA1, IB1, IA2, IB2, IX1, IY1, IY2, IX2, IY2, INTARY)

ARGUMENTS: (IA1, IB1) → (IA2, IB2): 1st line segment

(IX1, IY1) → (IX2, IY2): 2nd line segment

INTARY: array of length 5 set by WHEREX

$$\text{INTARY(1)} = \begin{cases} 0 & \text{No intersection} \\ 1 & \text{Intersection at a point} \\ 2 & \text{Line segments coincide} \end{cases}$$

$$\left.\begin{array}{l} \text{INTARY(2)} \\ \text{INTARY(3)} \end{array}\right\} \text{First Intersection Point}$$

$$\left.\begin{array}{l} \text{INTARY(4)} \\ \text{INTARY(5)} \end{array}\right\} \begin{array}{l} \text{Last Intersection} \\ \text{Point (set only if line segments coincide)} \end{array}$$

Supplement: Brief Description of Graphics Capabilities Used in TOMM

As described in Section 3.1.2, the TOMM display file is organized into a large number of independent entities, from several dozen to several hundred depending on the complexity of the scenario. Each entity may be independently moved on the screen, turned off and on, blinked, and changed in color. In fact, each entity may be redefined in any way without affecting the rest of the display. This capability is essential for the movement of units and cursors, and for the drawing and erasure of the many lines, contours, and messages used in TOMM. Entities may also be easily copied (this is useful for units).

At a lower level, the display capabilities implemented in hardware and used by TOMM include four colors and four intensities, line drawing of four types (solid, dotted, dashed, dot-dash), characters in four sizes, blinking of entities (or portions thereof), circle drawing, 32-key function keyboard, trackball, and lightpen (for reference lines only). Pressing one of the function keys causes a hardware interrupt as described in Section 3.1.3. The real-time clock on the host computer is used to time the replay of scenarios. The display area is approximately 1200 X 1600 points in resolution (an expanded 1024 X 1024 display).

RESEARCH NOTE 80-9

Dynamic Displays For Tactical Planning
Volume III: Software Documentation

Michael D. Schechterman and Lawrence R. Levi
Integrated Sciences Corporation

Human Factors Technical Area

DECEMBER 1979

This section
source code (not incl
package). The root s
order. Each overlay
overlay listings are
in nature or are used

U. S. Army
Research Institute for the Behavioral and Social Sciences

Approved for public release, di

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

7

cal Planning
umentation

ce R. Levi
ion

5.0 PROGRAM SOURCE LISTING

This section contains a complete listing of the 8400 lines of TOMM source code (not including system routines such as the FORTRAN graphics package). The root segment is listed first, followed by the 21 overlays in order. Each overlay is represented by its major subprograms. Following the overlay listings are a large number of subprograms which are either minor in nature or are used by several overlays.

Social Sciences

```
      IMPLICIT INTEGER (A-C,E-Z)
      LOGICAL DRTURN
      COMMON/DLAY/IDELY
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/DISPL/IDPL(4900),IERR
      COMMON/SCRTCH/PARRAY(81)
      COMMON ARRAY(63)
      COMMON/ATT/IATT(12)
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/FILES/FILNUM,FCB(13)
      COMMON/BRANCH/DRTURN
      COMMON/PROGRM/INTRPT
      COMMON/KEYLIT/ROW1,ROW2,ROW3,ROW4
      COMMON/ENTCNT/UNTENT,REFENT
      COMMON/UINFO/RCDSIZ
      COMMON/UINTL/ENDRNC(3,4),ASSETS(3,4)
      COMMON/SPEED/SPEEDS(4,9)
      COMMON/SCREEN/SIZE
      COMMON/SINGLE/UPTR,FILPTR
      COMMON/CLUSTR/UNTPTR(10,2),CLTYPE,XZERO,YZERO,XREF,YREF,UNTCNT
      COMMON/PROBLM/TIME
      COMMON/RPLAY/INDEX,INTRVL(4),ENDTIM
      COMMON/MOVE/MOVENT
      COMMON/SAVE/REFJ,FREE,JUMP

      DIMENSION IOVLAY(3,21)
      DATA IOVLAY(1,1),IOVLAY(2,1),IOVLAY(3,1)/'PA','GE','01'/
      DATA IOVLAY(1,2),IOVLAY(2,2),IOVLAY(3,2)/'PA','GE','02'/
      DATA IOVLAY(1,3),IOVLAY(2,3),IOVLAY(3,3)/'PA','GE','03'/
      DATA IOVLAY(1,4),IOVLAY(2,4),IOVLAY(3,4)/'PA','GE','04'/
      DATA IOVLAY(1,5),IOVLAY(2,5),IOVLAY(3,5)/'PA','GE','05'/
      DATA IOVLAY(1,6),IOVLAY(2,6),IOVLAY(3,6)/'PA','GE','06'/
      DATA IOVLAY(1,7),IOVLAY(2,7),IOVLAY(3,7)/'PA','GE','07'/
      DATA IOVLAY(1,8),IOVLAY(2,8),IOVLAY(3,8)/'PA','GE','08'/
      DATA IOVLAY(1,9),IOVLAY(2,9),IOVLAY(3,9)/'PA','GE','09'/
      DATA IOVLAY(1,10),IOVLAY(2,10),IOVLAY(3,10)/'PA','GE','10'/
      DATA IOVLAY(1,11),IOVLAY(2,11),IOVLAY(3,11)/'PA','GE','11'/
      DATA IOVLAY(1,12),IOVLAY(2,12),IOVLAY(3,12)/'PA','GE','12'/
      DATA IOVLAY(1,13),IOVLAY(2,13),IOVLAY(3,13)/'PA','GE','13'/
      DATA IOVLAY(1,14),IOVLAY(2,14),IOVLAY(3,14)/'PA','GE','14'/
      DATA IOVLAY(1,15),IOVLAY(2,15),IOVLAY(3,15)/'PA','GE','15'/
      DATA IOVLAY(1,16),IOVLAY(2,16),IOVLAY(3,16)/'PA','GE','16'/
      DATA IOVLAY(1,17),IOVLAY(2,17),IOVLAY(3,17)/'PA','GE','17'/
      DATA IOVLAY(1,18),IOVLAY(2,18),IOVLAY(3,18)/'PA','GE','18'/
      DATA IOVLAY(1,19),IOVLAY(2,19),IOVLAY(3,19)/'PA','GE','19'/
      DATA IOVLAY(1,20),IOVLAY(2,20),IOVLAY(3,20)/'PA','GE','20'/
      DATA IOVLAY(1,21),IOVLAY(2,21),IOVLAY(3,21)/'PA','GE','21'/

      CALL GINI(IDPL,4900,IATT,IERR)

10    CALL OVLAY(0,1,IOVLAY(1,OVLYKY(1)))
      IF(OVLYKY(1).NE.0)GOTO10

      STOP
      END
      BLOCK DATA
```

```
      IMPLICIT INTEGER (A-Z)
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/FILES/FILNUM,FCB(13)
      COMMON/KEYLIT/ROW1,ROW2,ROW3,ROW4
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/PROGRM/INTRPT
      COMMON/ENTCNT/ONTENT,PEFENT
      COMMON/OINFO/RCDSIZ
      COMMON/OINTL/ENDRNC(3,4),ASSETS(3,4)
      COMMON/SPEED/SPEEDS(4,9)
      COMMON/SCREEN/SIZE
      COMMON/PROBLM/TIME
      COMMON/RPLAY/INDEX,INTRVL(4),ENDTIM
      DATA MDF(3)/2000/
      DATA MDFMAX/3000/
      DATA OVLYKY/1,2,0,0,0/
      DATA FCB(3),FCB(8),FCB(9),FCB(10)/0,'SC','EN','ES'/
      DATA INTRPT/-1/
      DATA ROW1,ROW2,ROW3,ROW4/128,64,0,32/
      DATA PEFENT/300/
      DATA RCDSIZ/20/
      DATA ENDRNC/12*0/
      DATA ASSETS/12*0/
      DATA SPEEDS/25,25,4,25, 0,0,0,0, 0,0,0,0, 10,10,3,10,
     +   5,5,2,5, 5,5,2,5, 4,4,2,4, 10,10,3,10, 20,20,3,20/
      DATA SIZE/40/
      DATA TIME/0/
      DATA INDEX,INTRVL,ENDTIM/0,8,16,32,64,2880/
      END
END OF DATA
```

```
      SUBROUTINE C1
      CALL DRECTR
      RETURN
      END
END OF DATA
```

```
      SUBROUTINE DRECTR
      COMMON/MDFILE/MDF(3  0),MDFMAX
      COMMON/DISPL/IDPL(4900),IERR
      COMMON/KEYLIT/ROW1,ROW2,ROW3,ROW4
      COMMON/FILES/FILNUM,FCB(13)
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/BRANCH/DRTURN
      COMMON/PROGRM/INTRPT
      COMMON/PROBLM/PTIME
      COMMON/MOVE/MOVENT
      COMMON/SAVE/RFRJ,FREE,JUMP
      COMMON/SINGLE/UPTR,FILPTR
      IMPLICIT INTEGER (A-C,E-Z)
      INTEGER DFTER
      LOGICAL DRTURN

      DRTURN=.FALSE.
      UNITMB=3
      RANGCN=4
      DFTER=5
      USTATE=7
      LOSRGN=12
      REFLN=13
      UNTMMT=15
      AUDBE=20
      SETIM=21
      RCLSCN=22
      STPSCN=23
      RPSPD=27
      CURRPL=28
      MSTRPL=29
      XTCMM=31
      FILIN=1
      FILOUT=2

      IF(OVLYKY(2).EQ.0)GOTO1
      OVLYKY(1)=OVLYKY(2)
      OVLYKY(2)=OVLYKY(3)
      OVLYKY(3)=OVLYKY(4)
      OVLYKY(4)=OVLYKY(5)
      OVLYKY(5)=0
      RETURN
1     CONTINUE

2     CALL LAMPS(ROW1,ROW2,ROW3,ROW4)
      IF (INTRPT.EQ.-1) GO TO 3
      KEY=INTRPT
      INTRPT=-1
      GOTO35

3     CALL CKINT(KEY)

35    IF(KEY.NE.DFTER)GOTO4
      IF(ROW2.EQ.248)CALL SCINIT
      PTIME=0
```

```
      OVLYKY(1)=4
      OVLYKY(2)=5
      OVLYKY(3)=6
      ROW1=184
      ROW2=248
      ROW3=248
      ROW4=248
      RETURN

4     IF(KEY.NE.RCLSCN)GOTO5
      CALL SELFIL(FILIN)
      IF(DRTURN)GOTO2
      IF(ROW2.EQ.248)CALL SCINIT
      CALL SCRTER
      CALL SCRFOR
      CALL CLKUPD
      OVLYKY(1)=5
      ROW1=184
      ROW2=248
      ROW3=248
      ROW4=248
      RETURN

5     IF(KEY.NE.XTOMM)GOTO6
      OVLYKY(1)=0
      RETURN
6     IF(ROW2.EQ.96)GOTO3

      IF(KEY.NE.UNITMB)GOTO7
      CALL LITMOB
      GOTO2

7     IF(KEY.NE.STRSCN)GOTO8
      CALL SELFIL(FILOUT)
      GOTO2

8     IF(KEY.NE.REFLN)GOTO9
      OVLYKY(1)=3
      RETURN

9     IF(KEY.NE.USTATE)GOTO10
      OVLYKY(1)=7
      RETURN

10    IF(KEY.NE.RANGCN)GOTO11
      OVLYKY(1)=14
      OVLYKY(2)=15
      OVLYKY(3)=14
      OVLYKY(4)=16
      OVLYKY(5)=14
      RETURN

11    IF(KEY.NE.MSTRPL)GOTO12
      INTRPT=MSTRPL
      CALL LAMPS(72,0,0,0)
      OVLYKY(1)=21
```

```
      RETURN

12    IF(KEY.NE.CURRPL)GOTO13
      INTRPT=CURRPL
      UPTR=0
      FILPTR=0
      EVTPTR=MDF(4)
      IF (EVTPTR.GT.0) MDF(3)=MDF(EVTPTR)
      CALL LAMPS(72,0,0,0)
      CVLYKY(1)=21
      IF(PTIME.EQ.0)CVLYKY(1)=19
      RETURN

13    IF(KEY.NE.RPSPD)GOTO14
      CALL RPLSPD
      GOTO3

14    IF(KEY.NE.UNTMVT)GOTO15
      CALL GSAVE(I PL,FREQ,FREE,JUMP)
      EVTPTR=MDF(4)
      IF((EVTPTR.NE.0).AND.(PTIME.LT.MDF(EVTPTR+1)))MDF(EVTPTR+1)=PTIME
      IF(EVTPTR.NE.0)MDF(3)=MDF(EVTPTR)
      MCVENT=2968
      CVLYKY(1)=8
      RETURN

15    IF (KEY.NE.SETIM) GO TO 16
      CVLYKY(1)=20
      RETURN

   16 IF(KEY.NE.LCSPGN)GO TO 17
      CVLYKY(1)=17
      CVLYKY(2)=18
      RETURN

17    IF(KEY.NE.AOBE)GO TO 18
      CVLYKY(1)=17
      CVLYKY(2)=13
      RETURN

18    GO TO 3
      END

 END OF DATA
```

```
      SUBROUTINE SCRTER
      COMMON/MDFILE/MDF(3000),MDFMAX
      IMPLICIT INTEGER(A-Z)

C   ****  PUTS ALL TERRAIN IN MDF ONTO SCREEN.  SETS
C   ****  POINTER TO BEGINNING OF UNIT INFORMATION BLOCK

      FLPTR=1

1     FLPTR=MDF(FLPTR)
2     IF(MDF(FLPTR).EQ.0)GOTO4

      NXTYPE=MDF(FLPTR)
      TYPE=MDF(FLPTR+1)
      IF(TYPE.EQ.7)GOTO1

      ENT=MDF(FLPTR+2)
      FLPTR=FLPTR+4
      X=MDF(FLPTR+3)
      Y=MDF(FLPTR+4)
      CALL GBEG(ENT,X,Y)
      CALL TCOLOR(TYPE)
      CALL GPUT(5,1760,0,0)
      CALL GPUT(6,43,X,Y)

      EL=7
      FLPTR=FLPTR+5
3     IF(FLPTR.EQ.NXTYPE)GOTO2
      X=MDF(FLPTR)
      Y=MDF(FLPTR+1)
      CALL GPUT(EL,33,X,Y)
      EL=EL+1
      FLPTR=FLPTR+2
      GOTO3

4     MDF(2)=FLPTR+1

      RETURN
      END
 END OF DATA
```

```
      SUBROUTINE SCRFOR
      COMMON/MDFILE/MDF(5000),MDFMAX
      COMMON/UINFO/RCDSIZ
      IMPLICIT INTEGER(A-Z)
      DIMENSION ENTNUM(3,4)
      DATA ENTNUM/200,201,202,203,204,205,206,207,208,209,210,211/

C  ****  DISPLAYS FORCES AS DESCRIBED IN SCENARIO FILE

      UPTR=MDF(2)
      UTOTAL=MDF(UPTR)+MDF(UPTR+1)
      IF (UTOTAL.LE.0) RETURN
      EFLPTR=MDF(UPTR+3)

      FLPTR=UPTR+4

      RED=0
      SIDE=1

      DO 1 I=1,UTOTAL
             IF(FLPTR.EQ.EFLPTR)SIDE=2
             ENT=MDF(FLPTR+1)
             SIZE=MDF(FLPTR+2)
             TYPE=MDF(FLPTR+3)
             X=MDF(FLPTR+4)
             Y=MDF(FLPTR+5)
      CALL GCPY(ENT,ENTNUM(SIZE,TYPE),X,Y)
             IF(SIDE.EQ.2)CALL COLOR(RED)
      IF(SIDE.EQ.1)CALL GPUT(3,130,2,0)
      FLPTR=FLPTR+RCDSIZ
1     CONTINUE

      RETURN
      END
 END OF DATA
```

```
      SUBROUTINE LITMOB
      COMMON/BRANCH/DRTURN
      COMMON/DISPL/IDPL(4900),IERR
      IMPLICIT INTEGER (A-C,E-Z)
      LOGICAL DRTURN

C  ****  ROUTINE TURNS ON A SELECTED UNIT TYPE MOBILITY ILLUS.

2     CALL SELUTT(TYPE)
      CALL GHLT
      IF(ITSW(1))WRITE(14)IDPL
      CALL GSTT(0,0)

      DO 25 J=1,6
         CALL GEOF(2300+J)
25    CONTINUE

      IF(DRTURN)RETURN

      GOTO(3,4,5,6),TYPE

3     CALL GEON(2301)
      CALL GEON(2303)
      GOTO2

4     CALL GEON(2301)
      CALL GEON(2304)
      GOTO2

5     CALL GEON(2302)
      CALL GEON(2305)
      GOTO 2

6     CALL GEON(2301)
      CALL GEON(2306)
      GO TO 2

      END
 END OF DATA
```

```
      SUBROUTINE RPLSPD
      COMMON/RPLAY/INDEX,INTRVL(4),ENDTIM
      IMPLICIT INTEGER(A-Z)

C  ****  CHANGES REPLAY SPEED

      INDEX=INDEX+1
      IF(INDEX.EQ.5)INDEX=1

      RATE=200./FLOAT(INTRVL(INDEX))+0.5

      CALL GHLT

      CALL GSCH(4,35)
      WRITE(15,50)
50    FORMAT(' REPLAY COMPRESSION RATE')

      CALL GSCH(4,63)
      WRITE(15,100)RATE
100   FORMAT(I3,' MINUTES PER SECOND')

      CALL GSTT(0,0)

      RETURN
      END
 END OF DATA
```

```
      SUBROUTINE G2
      CALL SCINIT
      RETURN
      END
END OF DATA
```

```
      SUBROUTINE SCINIT
      COMMON/DISPL/IDPL(4900),IERR
      COMMON/ATT/IATT(12)
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/RPLAY/INDEX,INTRVL(4),ENDTIM
      IMPLICIT INTEGER (A-Z)

C   THIS HANDLES ALL DISPLAY ENTITY INITIALIZATION

      CALL GINI(IDPL,4900,IATT,IERR)
      CALL GSTT(0,0)

C   SET UP CURSOR

      CALL GBEG(1,0,0)
      CALL GEOF(1)
      CALL GPUT(3,130,2,0)
      CALL SHCCUR
      CALL GCPY(7,1,0,0)
      CALL GEOF(7)

      CALL GBEG(5,0,0)
      CALL GEOF(5)
      CALL GPUT(3,130,2,1)
      CALL COLOR(1)
      CALL GPUT(6,1000,25,0)

C   SET UP BORDER

      CALL GBEG(2,0,0)
      CALL GPUT(3,130,2,3)
      CALL GPUT(4,140,5,0)
      CALL GPUT(4,140,6,0)
      CALL GPUT(5,1740,0,1)
      CALL GPUT(6,51,1023,0)
      CALL GPUT(7,52,1023,0)
      CALL GPUT(8,51,-1023,0)
      CALL GPUT(9,52,-1023,0)

C   SET UP CHARACTER AREAS

      DO 10 I=0,40
      ENTITY=1000+I
      Y=1000-50*I
      CALL GBEG(ENTITY,-380,Y)
      CALL GEOF(ENTITY)
      CALL GPUT(4,140,5,0)
      CALL GPUT(4,140,6,0)
      CALL GPUT(5,1750,1,1)
      CALL GCHA(ENTITY,6,0,1,25)
10    CONTINUE

      CALL GENT(1000)
      CALL GPUT(3,130,2,1)
      CALL GPUT(4,140,5,0)
```

```
      CALL GPUT(4,140,0,1)
      CALL GPUT(32,100,-380,0)
      CALL GPUT(33,11 ,1000,0)
      CALL GPUT(34,114,50,0)
      CALL GCHA(1000,35,0,2,25)

C  SET UP MOVIE TIME THERMOMETER

      LOOP=9
      IDELTA=1024/(LOOP-1)
      CALL GBEG(3,500,500)
      CALL GPUT(3,130,2,0)
      CALL GPUT(5,1740,0,1)
      CALL GPUT(6,73,524,533)
      CALL GPUT(7,51,-1023,0)
      CALL GPUT(8,51,-1,0)
      CALL GPUT(9,72,5,0)
      IEL=10

      DO 20 I=1,LOOP
      CALL GPUT(IEL,52,-10,0)
      IEL=IEL+1
      CALL GPUT(IEL,73,IDELTA,10)
      IEL=IEL+1
20    CONTINUE

C  SET UP TIME POINTER

      CALL GBEG(4,500,500)
      CALL GPUT(3,130,2,0)
      CALL GPUT(5,1740,0,1)
      CALL GPUT(6,104,-500,0)
      CALL GPUT(7,114,568,0)
      CALL GPUT(8,52,-20,0)
      CALL GPUT(9,53,-5,10)
      CALL GPUT(10,71,10,0)
      CALL GPUT(11,53,-5,-10)
      CALL GPUT(12,73,-42,25)
      CALL GPUT(13,1750,2,1)
      CALL GPUT(14,90,1H0,0)
      CALL GPUT(15,90,1H0,0)
      CALL GPUT(16,90,1H0,0)
      CALL GPUT(17,90,1H0,0)
      CALL GPUT(18,104,-84,0)
      CALL GPUT(19,114,25,0)
      DO 21 EL=20,22
21    CALL GPUT(EL,90,1H ,0)
      CALL GPUT(33,100,0,0)
      CALL GPUT(34,114,225,0)
      CALL GCHA(4,35,0,1,25)
      CALL GPUT(61,114,-25,0)
      CALL GPUT(62,104,-350,0)
      CALL GCHA(4,63,0,1,25)
      IF (INDEX.GT.0) INDEX=INDEX-1
      CALL RPLSPD
```

```
C   ****   DRAW UNITS

       UNTENT=200
       DO 30 TYPE=1,4
              DO 25 SIZE=1,3
              CALL GBEG(UNTENT,0,0)
              CALL GEOF(UNTENT)
              EL=5
              CALL FORCES(SIZE,TYPE,EL)
             UNTENT=UNTENT+1
25            CONTINUE
30     CONTINUE

C   ****   MOBILITY TABLE

       CALL MBTBL

C   ****   CLUSTER CIRCLES

       DO 40 J=0,9
       CALL GCPY(8+J,5,0,0)
       CALL COLOR(1)
40     CALL GEOF(8+J)

       EVLYKY(1)=1

       RETURN
       END
  END OF DATA
```

```
      SUBROUTINE O3
      CALL REFLIN
      RETURN
      END
 END OF DATA
```

```
      SUBROUTINE REFLIN
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/BRANCH/DRTURN
      IMPLICIT INTEGER (A-C,E-Z)
      LOGICAL DRTURN

C  ****  ALLOWS USER TO MAINTAIN A SYSTEM OF REFERENCE LINES

      OVLYKY(1)=1

C  ****  ERASE OR CONSTRUCT LINES?  PROMPT USER
      CALL SELREF(PRCORE)
      IF(DRTURN)RETURN

C  **** DISPLAY APPROPRIATE MESSAGE
      CALL REFMSG(PRCORE)
      CALL UNSEL

      GOTO(1,2),PRCORE

1     CALL ADDREF
      CALL OFFSEL
      CALL BLKSEL
      RETURN

2     CALL DELREF
      CALL OFFSEL
      CALL BLKSEL
      RETURN

      END
 END OF DATA
```

```
      SUBROUTINE ADDREF
      COMMON/ENTCNT/CNTENT,REFENT
      IMPLICIT INTEGER (A-Z)

C   ****  ADDS USER DRAWN REFERENCE LINES TO THE SCREEN DISPLAY

      ENTITY=REFENT
      ACCEPT=1
      RTURN=30
      CURSOR=1
      DEVICE=1

      CALL LAMPS(64,0,0,2)

1     CALL GBEG(ENTITY,500,500)

      CALL GPUT(3,130,1,3)
      CALL GPUT(4,140,5,1)
      CALL GPUT(4,140,6,0)
      CALL GPUT(5,1760,0,0)

      CALL GEON(CURSOR)
2     CALL MOVENT(ENTITY,CURSOR,DEVICE,X,Y)
      CALL CKINT(KEY)
      IF(KEY.NE.ACCEPT.AND.KEY.NE.RTURN)GOTO2

      IF(KEY.NE.ACCEPT)GOTO5

      CALL GPUT(1,100,X,0)
      CALL GPUT(2,110,Y,0)
      CALL GPUT(6,1600,10,0)

      ELEMNT=7

3     CALL MOVENT(ENTITY,CURSOR,DEVICE,X,Y)
      CALL GPUT(ELEMNT,53,X,Y)
      CALL CKINT(KEY)
      IF(KEY.NE.ACCEPT.AND.KEY.NE.RTURN)GOTO3

      IF(KEY.NE.ACCEPT)GOTO4

      ELEMNT=ELEMNT+1
      GOTO3

4     CALL GPUT(ELEMNT+1,53,X,Y)
      CALL GPUT(ELEMNT+2,1600,10,0)
      ENTITY=ENTITY+1
      GOTO1

5     REFENT=ENTITY
      CALL GEOF(CURSOR)
      RETURN
      END
```

```
      SUBROUTINE DELREF
      COMMON/ENTCNT/UNTENT,REFENT
      COMMON/BRANCH/DRTURN
      IMPLICIT INTEGER(A-C,E-Z)
      LOGICAL DRTURN

      ENTNUM=300
      RANGE=REFENT-1
      ON=1
      OFF=0

      CALL LPSENS(ON,ENTNUM,RANGE)

1     CALL PENPIK(ENTITY)
      IF(DRTURN)GOTO2
      CALL GEOF(ENTITY)
      GOTO1

2     CALL LPSENS(OFF,ENTNUM,RANGE)

      RETURN
      END
```

```
SUBROUTINE O4
CALL DEFTER
RETURN
END
```

```
      SUBROUTINE DEFTER
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/BRANCH/OFTURN
      LOGICAL OFTURN
      INTEGER OVLYKY

C  DEFINES TERRAIN FEATURES INTO MDF AND SETS POINTER
C     INDEXING ADDRESS AFTER TERRAIN DATA

      DO 1 I=1,MDFMAX
         MDF(I)=0
1     CONTINUE

C  SET FUNCTION KEY VARIABLES

      IREJCT=0
      IACCPT=1

      IFLPTR=5
      MDF(1)=IFLPTR
      MDF(3)=2000
      MDF(4)=0
```

```
      IDEV=1
      IENT=101
C  UPDATE TERRAIN POINTER
10    ITPTR=IFLPTR
      IFLPTR=IFLPTR+1
C  DETERMINE WHICH TERRAIN TYPE HAS BEEN SELECTED
C     THEN ENTER IT INTO THE MASTER DATA FILE
      CALL SELTER(ITYPE)
      IF(LRTURN)GOTO50

      ICLASS=1
      IF(ITYPE .GE. 3) ICLASS=2

20    MDF(IFLPTR)=ITYPE
      IFLPTR=IFLPTR+1
      MDF(IFLPTR)=IENT
      IFLPTR=IFLPTR+1
      CALL GBEG(IENT,500,500)
      CALL TCOLOR(ITYPE)

      CALL DFTRMS(ICLASS)
      CALL TRENID(ITYPE)
      CALL CHSEL

      CALL SETPNT(IENT,IDEV,IX,IY,IACCPT)
      CALL GPUT(1,100,IX,0)
      CALL GPUT(2,110,IY,0)
      CALL GDRAW(ICLASS,MDF,IFLPTR,IENT,IX,IY)

      MDF(ITPTR)=IFLPTR

      IF(ITYPE.NE.5)GOTO28
      IHLPTR=ITPTR
      ITPTR=IFLPTR
      IFLPTR=IFLPTR+1
      ITYPE=8
      IENT=IENT+1
      GOTO20

28    IF(ITYPE.EQ.1)CALL ROAD(ITPTR)

      CALL BLKSEL

      CALL GHLT
      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('ACCEPT/REJECT FEATURE')
      CALL GSTT(0,0)

      CALL LAMPS(0,0,0,3)

30    CALL CKINT(KEY)
      IF(KEY .NE. IACCPT) GO TO 40
```

```
      IF(ITYPE.EQ.1)ITPTR=MDF(ITPTR)
      IFLPTR=MDF(ITPTR)
      IENT=MDF(ITPTR+2)+1
      GO TO 10

40    IF(KEY .NE. IREJCT) GO TO 30
      CALL GEOF(MDF(ITPTR+2))
      IENT=MDF(ITPTR+2)+1
      IFLPTR=ITPTR

      IF(ITYPE.NE.1)GOTO45
      CALL GEOF(MDF(ITPTR+2)+1)
      IENT=MDF(ITPTR+2)+2

45    IF(ITYPE.NE.6)GOTO10
      IFLPTR=IHLPTR
      CALL GEOF(MDF(IHLPTR+2))
      GOTO10

50    MDF(ITPTR)=0
      MDF(2)=ITPTR+1

      CALL OFFSEL
      CALL BLKSEL

      OVLYKY(1)=1

      RETURN
      END

 END OF DATA
```

```
      SUBROUTINE GDRAW(ITYPE,IARRAY,IPTR,IENT,JX,JY)

C   DRAWS SIMPLE PATHS OR CLOSED CONTOURS AND PUTS A
C      LIST OF THE ANCHOR POINTS INTO IARRAY

C              ITYPE  :  1    SIMPLE PATH
C                        2    CLOSED CONTOUR

C              IARRAY:  THE ARRAY TO CONTAIN THE ANCHOR POINTS

C              IPTR   :  INDEXES BEGINNING OF IARRAY

C              IENT   :  THE ENTITY NUMBER OF THE FIGURE

C              JX,JY  :  THE INITIAL POSITION OF THE FIGURE

      DIMENSION IARRAY(1)
      ITOP=IPTR
      IPTR=IPTR+4
      IXMIN=9999
      IXMAX=0
      IYMIN=9999
      IYMAX=0
      ICURSF=1
      IDEV=1
      TOLER=100
      INODE=1
      IFINIS=30
      KEY=-1
      NODTOT=0
      ITOPRW=0
      IF(ITYPE.EQ.1)ITOPRW=64
      CALL LAMPS(ITOPRW,0,0,2)
      CALL GSTT(0,0)
      CALL GCON(ICURSF)

C   SET TERRAIN TO ABSOLUTE MODE

      CALL GPUT(5,1760,0,0)
      IEL=6

C   SAVE STARTING POINT FOR LATER COMPARISON

      IX=JX
      IY=JY
      GO TO 30

20    CALL GPUT(IEL,53,IX,IY)
      IEL=IEL+1

C   PLACE NODE COORDINATES IN IARRAY

30    IARRAY(IPTR)=IX
```

```
      IARRAY(IPTR+1)=IY
      IF(IX .LT. IXMIN) IXMIN=IX
      IF(IY .LT. IYMIN) IYMIN=IY
      IF(IX .GT. IXMAX) IXMAX=IX
      IF(IY .GT. IYMAX) IYMAX=IY
      IPTR=IPTR+2

C  SHOW NODE ON SCREEN

      CALL GPUT(IEL,43,IX,IY)
      IEL=IEL+1
      NODTOT=NODTOT+1
      IF(KEY .EQ. IFINIS) GO TO 70

C  UPDATE RUBBER BAND SEGMENT FROM PREVIOUS NODE

40    CALL MOVENT(IENI,ICURSR,IDEV,IX,IY)
      CALL GPUT(IEL,53,IX,IY)
      CALL CKINT(KEY)
      IF(ITYPE .EQ. 1) GO TO 50

C  CHECK FOR CLOSURE AFTER SECOND NODE IS ESTABLISHED

      IF(NODTOT .LE. 1) GO TO 60
      DX=IX-JX
      DY=IY-JY
      DIST=DX*DX+DY*DY
      IF(DIST .GT. TOLER) GO TO 60

C  CLOSE THIS CONTOUR

      IX=JX
      IY=JY
      KEY=IFINIS
      GO TO 20

50    IF(KEY .EQ. IFINIS) GO TO 20

60    DX=IX-IARRAY(IPTR-2)
      DY=IY-IARRAY(IPTR-1)
      IF (DX*DX+DY*DY .LT. TOLER) GO TO 40
      IF (KEY-INODE) 40,20,40

C  STORE MIN, MAX COORDINATES

70    CALL GEOF(ICURSR)
      IARRAY(ITOP)=IXMIN
      IARRAY(ITOP+1)=IYMIN
      IARRAY(ITOP+2)=IXMAX
      IARRAY(ITOP+3)=IYMAX
      RETURN
      END
```

```
SUBROUTINE O5
CALL UNTMOB
RETURN
END
```

```
      SUBROUTINE UNTMOB
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/OVRLAY/OVLYKY(5)
      IMPLICIT INTEGER (A-R,T-Z)
      LOGICAL SCRNOT
      DIMENSION ENTNUM(2)
      DATA ENTNUM/2301,2302/

C   ****  PERFORMS ANALYSIS OF TERRAIN SO TO FACILLITATE MOBILITY ILLUS.

      OVLYKY(1)=1
      TPTR=MDF(1)
      IF (MDF(TPTR).EQ.0) RETURN

      ISKRPT=MDF(3)
      ININC=100
      SCRNOT=.FALSE.
      TYPEO=(-1)
      MASK=60
      LENGTH=MDFMAX-ISKRPT

      DO 1000 TYPE=1,2

      CALL GBEG(ENTNUM(TYPE),0,0)
      CALL GEOF(ENTNUM(TYPE))
      CALL COLOR(3)
      CALL GPUT(5,1760,0,0)
      ELEMNT=6

      X0=(-100)
      Y0=(-1125)
      X1=(1125)
      Y1=(100)
      LIMIT=925
      GOTO100

10    Y0=Y0+ININC
      Y1=Y1+ININC

      IF(Y0.LE.LIMIT)GOTO100
      IF(LIMIT.EQ.2150)GOTO1000

      Y0=(100)
      Y1=(-1125)
      LIMIT=2150

100   IF(SCRNOT)GOTO101
      TYPEO=IWRAMI(X0,Y0,MASK)
      CALL TTLEGS(X0,Y0,X1,Y1,TYPEO,MASK,MDF(ISKRPT),LENGTH)
```

```
      GOTO200
1 1   TYPEO=IWRAM!(X1,Y1,MASK)
      CALL TTLEGS(X1,Y1,X0,Y0,TYPEO,MASK,MDF(ISKRPT),LENGTH)
200   CALL DRTMOB(TYPE,MDF(ISKRPT),ELEMNT)
      SCRNOT=.NOT.SCRNOT
      GOTO10
1000  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DRTMOB(TYPE,IARRAY,ELEMNT)
      IMPLICIT INTEGER (A,C,D,F-R,U-Z)
      INTEGER TERRAN,TYPE,TTYPE,MOBIL,ELEMNT
      LOGICAL BEAMPS,TRANS,ENTER,EXIT,SETBIT
      DIMENSION MOBLTY(6,2),SETBIT(5),IARRAY(1)
      DATA MOBLTY/2,0,1,3,3,2,2,0,2,1,1,2/

C  ****  GENERATES GRAPHICS FOR MOBILITY ILLUSTRATIONS

      EXIT=.FALSE.
      ENTER=.TRUE.
      LMOBIL=99
      ROAD=1
      LAKE=2
      CITY=3
      FOREST=4
      HILL=5
      CLEAR=6
      DO 1 TERRAN=ROAD,HILL
         SETBIT(TERRAN)=.FALSE.
1     CONTINUE
      BEAMPS=.FALSE.
      PNTCNT=IARRAY(1)
      IF(PNTCNT.LE.0)RETURN

      DO 10 N=1,PNTCNT
          DELTAT=IARRAY(N*5+3)-IARRAY(N*5-2)
         TRANS=EXIT
         IF(DELTAT.GT.0)TRANS=ENTER
         TTYPE=IABS(DELTAT)/4+1
         IF(TTYPE.EQ.9)TTYPE=4
         SETBIT(TTYPE)=TRANS
         IF(DELTAT.EQ.8.AND.SETBIT(LAKE))LAKE=CITY
         IF(DELTAT.EQ.(-8).AND.SETBIT(2))LAKE=2

         TERRAN=ROAD
         IF(SETBIT(TERRAN))GOTO120
         DO 110 TERRAN=LAKE,HILL
            IF(SETBIT(TERRAN))GOTO120
110      CONTINUE
         TERRAN=CLEAR
```

```
120            MOBIL=MOBLTY(TERRAN,TYPE)
               IF(MOBIL.NE.2)GOTO1210
               BEAMPS=.FALSE.
               GOTO10

1210           IF(BEAMPS)GOTO1220
               CALL GPUT(ELEMNT,73,IARRAY(N*5),IARRAY(N*5+1))
               ELEMNT=ELEMNT+1
1220           IF(MOBIL.EQ.LMOBIL)GOTO1221
               CALL GPUT(ELEMNT,130,1,MOBIL)
             IF (MOBIL.EQ.1) CALL GPUT(ELEMNT,130,2,3)
             IF (MOBIL.EQ.3) CALL GPUT(ELEMNT,130,2,2)
             IF (MOBIL.EQ.0) CALL GPUT(ELEMNT,130,2,2)
               LMOBIL=MOBIL
               ELEMNT=ELEMNT+1
1221           CALL GPUT(ELEMNT,53,IARRAY(N*5+5),IARRAY(N*5+6))
               ELEMNT=ELEMNT+1
               BEAMPS=.TRUE.
10    CONTINUE
      RETURN

      END
```

```
SUBROUTINE O6
CALL UNITS
RETURN
END
```

```
      SUBROUTINE UNITS
      COMMON/BRANCH/DRTURN
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/UINTL/ENDRNC(3,4),ASSETS(3,4)
      COMMON/UINFO/RCDSIZ
      COMMON/ENTCNT/UNTENT,REFENT
      IMPLICIT INTEGER (A-C,E-Z)
      LOGICAL DRTURN
      DIMENSION ENTNUM(3,4)
      DATA ENTNUM/200,201,202,203,204,205,206,207,208,209,210,211/
C **** ESTABLISHES INITIAL ORDER OF BATTLE
      SIDE=1
      TBALL=1
      ENT=240
      REJECT=0
      ACCEPT=1
      POSTUR=1
      UCOUNT=0
      FUCNT=0
      RED=0
```

```
      JLOOK=189

      UPTR=MDF(2)
      FLPTR=UPTR+4

1     CALL OOBMSG(SIDE)
      CALL SELUTT(TYPE)

      IF(.NOT.DRTURN)GOTO2
      IF(SIDE.EQ.2)GOTO6

      SIDE=2
      FUCNT=UCOUNT
      UCOUNT=0
      MDF(UPTR+3)=FLPTR
      GOTO1

2     IF (TYPE.NE.1) CALL SELUTS(SIZE)
      IF (TYPE.EQ.1) SIZE=3
      IF(DRTURN)GOTO1

      CALL GCPY(ENT,ENTNUM(SIZE,TYPE),500,500)
      IF(SIDE.EQ.2)CALL COLOR(RED)
      IF(SIDE.EQ.1)CALL GPUT(3,130,2,0)

      CALL LAMPS(0,0,0,3)

      CALL ADUMSG
      CALL ONSEL

3     CALL MOVENT(0,ENT,TBALL,X,Y)
      CALL CKINT(KEY)
      IF(KEY.NE.ACCEPT)GOTO5

      CALL OFFSEL
      CALL BLKSEL

      UCOUNT=UCOUNT+1

      MDF(FLPTR)=UCOUNT
      MDF(FLPTR+1)=ENT
      MDF(FLPTR+2)=SIZE
      MDF(FLPTR+3)=TYPE
      MDF(FLPTR+4)=X
      MDF(FLPTR+5)=Y
      MDF(FLPTR+6)=IWRAMI(X,Y,JLOOK)
      MDF(FLPTR+7)=0
      MDF(FLPTR+8)=ASSETS(SIZE,TYPE)
      MDF(FLPTR+9)=ENDRNC(SIZE,TYPE)
      DO 4 I=10,17
      MDF(FLPTR+I)=0
4     CONTINUE
      MDF(FLPTR+18)=POSTUR

      FLPTR=FLPTR+RCDSIZ
      ENT=ENT+1
```

```
      GOTO1

5     IF(KEY.NE.REJECT)GOTO3
      CALL GEOF(ENT)
      ENT=ENT+1
      CALL OFFSEL
      CALL BLKSEL
      GOTO1

6     EUCNT=UCOUNT
      MDF(UPTR)=FUCNT
      MDF(UPTR+1)=EUCNT
      MDF(UPTR+2)=UPTR+4
      MDF(3)=FLPTR
      UNTENT=ENT

      IF (FUCNT+EUCNT .NE. 0) CALL ACTBLK

      OVLYKY(1)=1

      RETURN
      END
```

```
      SUBROUTINE ACTBLK
      COMMON/MDFILE/MDF(3000),MDFMAX
       COMMON/UINFC/RCDSIZ
      COMMON/PROBLM/TIME
      COMMON/RPLAY/INDEX,INTRVL(4),ENDTIM
      IMPLICIT INTEGER (A-Z)

C  ****  SETS UP DATABASE FOR RECORDING OF UNIT ACTIONS

      UPTR=MDF(2)
      UCOUNT=MDF(UPTR)+MDF(UPTR+1)
      UPTR=UPTR+4
      SKRPTR=MDF(3)
      RESPLV=100
       POSTUR=1
C  ****  ACTIVE UNIT AT X,Y,DETECTION,ENGAGE.,FIRING RNGE.,RESUP.,POSTURE
       BTCODE=4095
C  ****  MOVE TO X,Y
      BTCOD2=3075

      DO 1 J=1,UCOUNT

      MDF(UPTR+RCDSIZ-1)=SKRPTR
      MDF(SKRPTR)=3
      MDF(SKRPTR+1)=3
      MDF(SKRPTR+2)=16
      MDF(SKRPTR+3)=BTCODE
      MDF(SKRPTR+4)=MDF(UPTR+4)
      MDF(SKRPTR+5)=MDF(UPTR+5)
      MDF(SKRPTR+6)=TIME
      MDF(SKRPTR+7)=MDF(UPTR+6)
```

```
      DO 10 I=8,13
      MDF(SKRPTR+I)=0
10    CONTINUE
      MDF(SKRPTR+14)=RESPLV
      MDF(SKRPTR+15)=POSTUR
      MDF(SKRPTR+16)=BTCOD2
      MDF(SKRPTR+17)=MDF(UPTR+4)
      MDF(SKRPTR+18)=MDF(UPTR+5)
      MDF(SKRPTR+19)=ENDTIM
      MDF(SKRPTR+20)=MDF(UPTR+6)

       UPTR=UPTR+RCDSIZ
      SKRPTR=SKRPTR+21

1     CONTINUE

      MDF(3)=SKRPTR

      RETURN
      END
```

```
SUBROUTINE O7
CALL USTATE
RETURN
END
```

```
      SUBROUTINE USTATE
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/BRANCH/ORTURN
      IMPLICIT INTEGER (A-Z)

      IF(MDF(4).EQ.0)CALL EVTINT
1     CALL SELUSP(PRCODE)
      IF(ORTURN)GOTO100

      GOTO(10,20,30,40,50),PRCODE

10    CALL ADDUNT
      GOTO1

20    CALL DELUNT
      GOTO1

30    CALL RSPUNT
      GOTO1

40    CALL DEFMSN
      GOTO1

50    CALL PLOUNT
      GOTO1
```

```
1.0   OVLYKY(1)=1

      RETURN
      END


      SUBROUTINE ADDUNT
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/BRANCH/DRTURN
      COMMON/PROBLM/TIME
      COMMON/ENTCNT/UNTENT,REFENT
      COMMON/UINTL/ENGRNC(3,4),ASSETS(3,4)
      COMMON/UINFC/RCDSIZ
      COMMON/FPLAY/INDEX,INTRVL(4),ENDTIM
      IMPLICIT INTEGER(A-C,E-Z)
      LOGICAL DRTURN
      DIMENSION ENTNUM(3,4)
      DATA ENTNUM/200,201,202,203,204,205,206,207,208,209,210,211/

C  ****  ROUTINE PERFORMS DATABASE OPERATIONS FOR ADDING UNITS

      TBALL=1
      ACCEPT=1
      REJECT=0
      RED=0
      ENT=UNTENT
      UPTR=MDF(2)
      FUCNT=MDF(UPTR)
      EUCNT=MDF(UPTR+1)
      UCOUNT=EUCNT+FUCNT
      FFLPTR=MDF(UPTR+2)
      EFLPTR=MDF(UPTR+3)
       POSTUR=1
      JLOOK=189
C  ****  ACTIVE UNIT AT X,Y, DETECTION,ENGAGE.,FIRING RNGE.,RESUP.,POSTURE
       BTCODE=4095
C  ****  MOVE TO X,Y
      BTCOD1=3075
C  ****  INACTIVE UNIT AT X,Y
      BTCOD2=14337
      RESPLV=100

1      CALL USTMSG(1)
      CALL SELUTT(TYPE)
      IF (DRTURN) RETURN

2     IF (TYPE.NE.1) CALL SELUTS(SIZE)
      IF (TYPE.EQ.1) SIZE=3
      IF(DRTURN)GOTO1

3     CALL SELSID(SIDE)
      IF(DRTURN)GOTO1

      CALL GCPY(ENT,ENTNUM(SIZE,TYPE),500,500)
```

```
      IF(SIDE.EQ.2)CALL COLOR(RED)
      IF(SIDE.EQ.1)CALL GPUT(3,130,2,0)

      CALL LAMPS(0,0,0,3)
      CALL ADDMSG
      CALL UNSEL

4     CALL MOVENT(0,ENT,TBALL,X,Y)
      CALL CKINT(KEY)

      IF(KEY.NE.ACCEPT)GOTO11
      CALL OFFSEL
      CALL BLKSEL

      TTYPE=IWRAMI(X,Y,JLOOK)

C  ****  SHIFT THE MDF TO MAKE ROOM FOR NEW UNIT
      START=FFLPTR+UCOUNT*RCDSIZ
      IF(SIDE.EQ.1)START=EFLPTR
      TOTAL=RCDSIZ
      CALL MDFSHF(TOTAL,START)
      IF (SIDE.EQ.1) MDF(UPTR+3)=START+TOTAL

      EFLPTR=MDF(UPTR+3)
      SKRPTR=MDF(3)
      EVTPTR=MDF(4)

C  ****  INSERT UNIT INFORMATION RECORD
      FLPTR=START
      MDF(FLPTR)=1
      IF (SIDE.EQ.1 .AND. FUCNT.EQ.0) GO TO 5
      IF (SIDE.EQ.2 .AND. EUCNT.EQ.0) GO TO 5
      MDF(FLPTR)=MDF(FLPTR-RCDSIZ)+1
5     CONTINUE
      MDF(FLPTR+1)=ENT
      MDF(FLPTR+2)=SIZE
      MDF(FLPTR+3)=TYPE
      MDF(FLPTR+4)=X
      MDF(FLPTR+5)=Y
      MDF(FLPTR+6)=TTYPE
      MDF(FLPTR+7)=0
      MDF(FLPTR+8)=ASSETS(SIZE,TYPE)
      MDF(FLPTR+9)=ENDRNC(SIZE,TYPE)
      DO 15 I=10,17
         MDF(FLPTR+I)=0
15    CONTINUE
      MDF(FLPTR+18)=POSTUR

C  ****  RED UNIT ACTION RECORD GOES AT SCRATCH AREA BEGINNING
      IF(SIDE.EQ.1)GOTO6
      START=EVTPTR
      GOTO8

C  ****  GREEN UNIT ACTION RECORD GOES AFTER LAST GREEN UNIT ACTION RECORD
6     IF (EUCNT.NE.0) GO TO 7
      START=EVTPTR
```

```
      GO TO 8
C  ****  SHIFT MDF TO MAKE ROOM IN ACTION BLOCK FOR GREEN ACTION RECORD
7     PTRPTR=EFLPTR+RCDSIZ-1
      ACTPTR=MDF(PTRPTR)
      START=ACTPTR
8     TOTAL=21
      IF(TIME.NE.0)TOTAL=26

      IF(SIDE.EQ.1)FUCNT=FUCNT+1
      IF(SIDE.EQ.2)EUCNT=EUCNT+1
      UCCUNT=UCOUNT+1
      ENT=ENT+1
      MDF(UPTR)=FUCNT
      MDF(UPTR+1)=EUCNT
      UNTENT=ENT

      CALL MDFSHF(TOTAL,START)

      MDF(FLPTR+19)=START

      FLPTR=START

      IF(TIME.NE.0)GOTO9

C  ****  INSERT ACTION RECORD FOR A UNIT ADDED AT PROBLEM TIME ZERO
      MDF(FLPTR)=3
      MDF(FLPTR+1)=3
      MDF(FLPTR+2)=16
      MDF(FLPTR+3)=BTCODE
      MDF(FLPTR+4)=X
      MDF(FLPTR+5)=Y
      MDF(FLPTR+6)=TIME
      MDF(FLPTR+7)=TTYPE
      DO 18 I=8,13
         MDF(FLPTR+I)=0
18    CONTINUE
      MDF(FLPTR+14)=RESPLV
      MDF(FLPTR+15)=POSTUR
      MDF(FLPTR+16)=BTCOD1
      MDF(FLPTR+17)=X
      MDF(FLPTR+18)=Y
      MDF(FLPTR+19)=ENDTIM
      MDF(FLPTR+20)=TTYPE

      GO TO 1

C  ****  ACTION RECORD FOR UNIT ADDED AT OTHER THAN PROBLEM TIME ZERO
9     MDF(FLPTR)=7
      MDF(FLPTR+1)=7
      MDF(FLPTR+2)=21
      MDF(FLPTR+3)=BTCOD2
      MDF(FLPTR+4)=X
      MDF(FLPTR+5)=Y
      MDF(FLPTR+6)=0
      MDF(FLPTR+7)=TTYPE
      MDF(FLPTR+8)=BTCODE
```

```
      MDF(FLPTR+9)=X
      MDF(FLPTR+10)=Y
      MDF(FLPTR+11)=TIME
      MDF(FLPTR+12)=TTYPE
      DO 10 I=13,18
          MDF(FLPTR+I)=0
10    CONTINUE
      MDF(FLPTR+19)=RESPLV
      MDF(FLPTR+20)=POSTUR
      MDF(FLPTR+21)=BTCOD1
      MDF(FLPTR+22)=X
      MDF(FLPTR+23)=Y
      MDF(FLPTR+24)=ENDTIM
      MDF(FLPTR+25)=TTYPE

      GO TO 1

11    IF(KEY.NE.REJECT)GOTO4
      CALL GEOF(ENT)
      ENT=ENT+1
      CALL OFFSEL
      CALL BLKSEL
      GOTO1

      END
```

```
      SUBROUTINE DELUNT
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/BRANCH/DRTURN
      COMMON/PROBLM/TIME
      COMMON/UINFO/RCDSIZ
      IMPLICIT INTEGER (A-Z)
      LOGICAL DRTURN

C **** ROUTINE PERFORMS DATABASE OPERATIONS FOR DELETING UNITS
      IF (TIME.GT.0) RETURN

      UPTR=MDF(2)
      UCOUNT=MDF(UPTR)+MDF(UPTR+1)
      IF (UCOUNT.EQ.0) RETURN
      U1PTR=UPTR+4
10    ACTBLK=MDF(U1PTR+RCDSIZ-1)
      EFLPTR=MDF(UPTR+3)
      EVTPTR=MDF(4)

      CALL USTMSG(2)
      CALL UNSEL
      CALL SELUNT(UNTPTR)
      IF (DRTURN) GO TO 50
      ENTITY=MDF(UNTPTR+1)
      CALL GEOF(ENTITY)
      SIDE=1
      IF (UNTPTR.GE.EFLPTR) SIDE=2
```

```
C **** DELETE UNIT ACTION RECORDS

      START=MDF(UNTPTR+RCDSIZ-1)
      END=MDF(UNTPTR+2*RCDSIZ-1)
      IF (UNTPTR+RCDSIZ.GE.ACTBLK) END=EVTPTR
      TOTAL=START-END
      CALL MDFSHF(TOTAL,END)

C **** DELETE UNIT INFORMATION RECORD

      END=UNTPTR+20
      TOTAL=-RCDSIZ
      CALL MDFSHF(TOTAL,END)

      IF (SIDE.EQ.1) MDF(UPTR)=MDF(UPTR)-1
      IF (SIDE.EQ.2) MDF(UPTR+1)=MDF(UPTR+1)-1
      UCOUNT=UCOUNT-1
      IF (UCOUNT.EQ.0) GO TO 50
      GO TO 10

50    CALL OFFSEL
      CALL BLKSEL
      RETURN
      END
```

```
      SUBROUTINE RSPUNT
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/BRANCH/DRTURN
      COMMON/PROBLM/TIME
      COMMON/UINFO/RCDSIZ
      COMMON/ENTCNT/UNTENT,REFENT
      IMPLICIT INTEGER(A-C,E-Z)
      LOGICAL DRTURN
C   ****  ROUTINE PERFORMS DATABASE OPERATIONS FOR RESUPPLYING UNITS
C   ****  ACTIVE UNIT AT X,Y, RESUPPLIED WITH NEW COMBAT POSTURE
     BTCOD1=3841
      POSTUR=1
      UNTPTR=MDF(2)
     FFLPTR=MDF(UNTPTR+3)
     ACTBLK=MDF(FFLPTR+RCDSIZ-1)
      SKRPTR=MDF(3)
1    CALL USTMSG(3)
     CALL ONSEL
     CALL SELUNT(UPTR)
     CALL OFFSEL
     CALL BLKSEL
     IF(DRTURN)GOTO4
     ENTITY=MDF(UPTR+1)
C   ****  NEW UNIT POSTURE RECORDED IN UNIT INFORMATION RECORD
     MDF(UPTR+18)=POSTUR
```

```
C    ****  SHIFT MDF TO MAKE SPACE FOR THE RESUPPLY VECTOR
          ACTPTR=MDF(UPTR+RCDSIZ-1)
           PRESNT=MDF(ACTPTR+1)
           START=ACTPTR+PRESNT
           TOTAL=6
          CALL MDFSHF(TOTAL,START)

          SKRPTR=MDF(3)

C    ****  ALLOW USER TO INPUT A RESUPPLY LEVEL FROM THE FUNCTION KEYBOARD
3          CALL RSPMSG
           CALL UNSEL
           CALL SELNUM(RESPLV)
          IF(URTURN)GOTO1
           CALL OFFSEL
           CALL BLKSEL
           IF(RESPLV.GT.100)GOTO3

           CALL GENT(ENTITY)
           CALL GPUT(3,130,1,0)

C    ****  INSERT THE RESUPPLY VECTOR  (ALSO INCLUDED IS A NEW COMBAT POSTURE)
           MDF(ACTPTR+PRESNT)=BTCOD1
           MDF(ACTPTR+PRESNT+1)=MDF(UPTR+4)
           MDF(ACTPTR+PRESNT+2)=MDF(UPTR+5)
           MDF(ACTPTR+PRESNT+3)=TIME
           MDF(ACTPTR+PRESNT+4)=RESPLV
           MDF(ACTPTR+PRESNT+5)=POSTUR

C    ****  UPDATE UNIT ACTION RECORD RELATIVE POINTERS
           MDF(ACTPTR)=PRESNT
           MDF(ACTPTR+1)=PRESNT+6
           MDF(ACTPTR+2)=MDF(ACTPTR+2)+6

           GOTO1

4         RETURN

           END
```

```
      SUBROUTINE DEFMSN
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/BRANCH/DRTURN
      COMMON/PROBLM/TIME
      COMMON/UINFO/RCDSIZ
      COMMON/ENTCNT/UNTENT,REFENT
      IMPLICIT INTEGER (A-C,E-Z)
      LOGICAL DRTURN
C **** ROUTINE PERFORMS DATABASE OPERATIONS FOR DEFINING UNIT MISSION
C **** ACTIVE UNIT AT X,Y, WITH NEW COMBAT POSTURE
      BTCODE=3585
```

```
      UNTPTR=MDF(2)
      FELPTR=MDF(UNTPTR+3)
      ACTBLK=MDF(FELPTR+RCDSIZ-1)
      SKRPTR=MDF(3)

1     CALL USTMSG(4)
      CALL UNSEL
      CALL SELUNT(UPTR)
      CALL OFFSEL
      CALL BLKSEL
      IF(DRTURN)GOTO3

C **** SHIFT THE MDF TO MAKE SPACE FOR THE RESUPPLY VECTOR
      ACTPTR=MDF(UPTR+RCDSIZ-1)
      PRESNT=MDF(ACTPTR+1)
      START=ACTPTR+PRESNT
      TOTAL=5
      CALL MDFSHF(TOTAL,START)

      SKRPTR=MDF(3)

C **** ALLOW USER TO SELECT A MISSION (COMBAT POSTURE)
      CALL SELPOS(POSTUR)
      IF(DRTURN)GOTO1

C **** INSERT THE NEW COMBAT POSTURE VECTOR
      MDF(ACTPTR+PRESNT)=BTCCDE
      MDF(ACTPTR+PRESNT+1)=MDF(UPTR+4)
      MDF(ACTPTR+PRESNT+2)=MDF(UPTR+5)
      MDF(ACTPTR+PRESNT+3)=TIME
      MDF(ACTPTR+PRESNT+4)=POSTUR

C **** UPDATE UNIT ACTION RECORD RELATIVE POINTERS
      MDF(ACTPTR)=PRESNT
      MDF(ACTPTR+1)=PRESNT+5
      MDF(ACTPTR+2)=MDF(ACTPTR+2)+5
      GOTO1

3     RETURN

      END
```

```
      SUBROUTINE RLOUNT
      COMMON/MOFILE/MDF(3000),MDFMAX
      COMMON/BRANCH/DRTURN
      COMMON/PROBLM/TIME
      COMMON/UINFO/RCDSIZ
      COMMON/ENTCNT/UNTENT,REFENT
      IMPLICIT INTEGER(A-C,E-Z)
      LOGICAL DRTURN
C **** ROUTINE PERFORMS DATABASE OPERATIONS FOR RELOCATING UNIT'S POSITION
```

```
      JLOCK=169
      ACCEPT=1
      RTURN=30
      TBALL=1
      UNTPTR=MDF(2)
      FFLFTR=UNTPTR+4
      ACTBLK=MDF(FFLPTR+RCDSIZ-1)
      SKRPTR=MDF(3)
      ENDTIM=1440
C ****  INACTIVE UNIT AT X,Y
      BTCODE=2049

1     CALL USTMSG(5)
      CALL UNSEL
      CALL SELUNT(UPTR)
      CALL OFFSEL
      CALL BLKSEL
      IF(DRTURN)GOTO6
      ENTITY=MDF(UPTR+1)
      CALL LAMPS(64,0,0,2)

C ****  DETERMINE IF RELOCATE WILL PREEMTP OTHER ACTIONS
      ACTPTR=MDF(UPTR+RCDSIZ-1)
      IF(MDF(ACTPTR+19).NE.ENDTIM)CALL WARNMG
      CALL ADDMSG
      CALL UNSEL

2     CALL MOVENT(0,ENTITY,TBALL  ,X,Y)
      CALL CKINT(KEY)

      IF(KEY.NE.ACCEPT)GOTO5
C ****  UPDATE POSITION IN UNIT INFORMATION RECORD
      MDF(UPTR+4)=X
      MDF(UPTR+5)=Y
      MDF(UPTR+6)=IWRAMI(X,Y,JLOCK)

C ****  UPDATE INITIAL POSITION OF UNIT AS RECORDED IN ACTION VECTORS
      MDF(ACTPTR+4)=X
      MDF(ACTPTR+5)=Y
      MDF(ACTPTR+7)=MDF(UPTR+6)
      MDF(ACTPTR+16)=BTCODE
      MDF(ACTPTR+17)=X
      MDF(ACTPTR+18)=Y
      MDF(ACTPTR+19)=ENDTIM

C ****  FINAL RECORD IN THE ACTION BLOCK--JUST UPDATE THE SCRATCH POINTER
      IF(UPTR+RCDSIZ.NE.ACTBLK)GOTO3
      SKRPTR=ACTPTR+20
      CALL OFFSEL
      CALL BLKSEL
      GOTO1

C ****  SHIFT THE MDF TO THE RELOCATE VECTOR  (USE BLANK SPACE)
3     UPTR=UPTR+RCDSIZ
      START=MDF(UPTR+RCDSIZ-1)
```

```
      TOTAL=ACTPTR+20-START
      CALL MDFSHF(TOTAL,START)

      SKRPTR=MDF(3)
      CALL OFFSEL
      CALL BLKSEL
      GOTO1

C  ****  USER REJECTS RELOCATE
5     IF(KEY.NE.RTURN)GOTO2
      CALL GPUT(1,100,MDF(UPTR+4),0)
      CALL GPUT(2,110,MDF(UPTR+5),0)
      CALL OFFSEL
      CALL BLKSEL
      GOTO1

6     CALL OFFSEL
      CALL BLKSEL

      RETURN
      END

 END OF DATA
```

```
SUBROUTINE 08
CALL MVMMI
RETURN
END
```

```
      SUBROUTINE MVMMI
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/DISPL/IDPL(4900),IERR
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/BRANCH/DRTURN
      COMMON/SINGLE/UPTR,FILPTR
      COMMON/CLUSTR/UNTPTR(10,2),CLTYPE,XZERO,YZERO,XREF,YREF,UNTCNT
      COMMON/MOVE/MOVENT
      COMMON/SAVE/RFRJ,FREE,JUMP
      IMPLICIT INTEGER(A-C,E-Z)
       COMMON ARRAY(63)
      LOGICAL DRTURN

C  ****  ALLOWS USER TO SPECIFY OPTIONS IN PLANNING UNIT MOVEMENT

      OVLYKY(2)=9
      OVLYKY(3)=10
      OVLYKY(4)=11

C **** CREATE MOVEMENT PATH AND ARRIVAL TIME ENTITIES

      MOVENT=MOVENT+32
      CALL GBEG(MOVENT,0,0)
      CALL COLOR(2)
      CALL GPUT(5,1760,0,0)
      DO 5 I=1,10
          CALL GBEG(MOVENT+I,0,0)
          CALL COLOR(7)
          CALL GCHA(MOVENT+I,5,0,1,4)
5     CONTINUE

C  ****  SELECT TYPE OF UNIT MOVEMENT

10    CALL SELMVT(TYPE)
      IF(.NOT.DRTURN)GOTO12
      CALL GREST(IDPL,RFRJ,FREE,JUMP)
      DO 11 I=2,5
11    OVLYKY(I)=0
      GOTO50

12    IF(CLTYPE.EQ.1.OR.TYPE.NE.6)GOTO20
      DO 15 I=1,UNTCNT
           CALL GENT(7+I)
           CALL GPUT(1,100,MDF(UNTPTR(I,1)+4),0)
           CALL GPUT(2,110,MDF(UNTPTR(I,1)+5),0)
           CALL GEON(7+I)
15    CONTINUE
```

```
      UPTR=0
      GOTO40

C  ****  SELECT UNIT(S)

20    IF(TYPE.NE.1)GOTO30
      CALL SELUNT(UPTR)
      IF(URTURN)GOTO10
      CLTYPE=1
      GOTO40
30    CLTYPE=TYPE
      UPTR=0
      CALL IOUNTS
      IF(UNTCNT.EQ.0)GOTO10

C  ****  LOCATE CLUSTER CENTER

      IF(TYPE.NE.2.AND.TYPE.NE.3)GOTO35
      CALL CLCMSG
      CALL UNSEL
      CALL SETPNT(0,1,XZERO,YZERO,1)
      CALL OFFSEL
      CALL BLKSEL

C  ****  SELECT A REFERENCE POINT

      IF(TYPE.NE.3)GOTO40
      CALL REPTMG
      CALL UNSEL
      CALL SETPNT(0,1,XREF,YREF,1)
      CALL OFFSEL
      CALL BLKSEL
      GOTO40

C  ****  LEAD UNIT OF COLUMN OR LEAPFROG

35    XZERO=MDF(UNTPTR(1,1)+4)
      YZERO=MDF(UNTPTR(1,1)+5)

C  ****  INPUT THE PATH OF THE MOVEMENT

40    CALL DRWPTH
      IF(ARRAY(1).NE.0)GOTO50
      DO 45 I=0,9
45    CALL GECF(8+I)
      GOTO10

50    OVLYKY(1)=1
      RETURN

      END
```

```
      SUBROUTINE ~~SUP~~ IDUNTS
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/BRANCH/DRTURN
      COMMON/CLUSTR/UNTPTR(10,2),CLTYPE,XZERO,YZERO,XREF,YREF,UNTCNT
      IMPLICIT INTEGER (A-C,E-Z)
      LOGICAL DRTURN
      DIMENSION TABLE(5)
      DATA TABLE/1,10,10,10,2/

C  ****  ALLOWS USER TO CONSTRUCT A CLUSTER OF UNITS

      ON=1
      OFF=2
      UNTCNT=0
      UNTMAX=TABLE(CLTYPE)
      DO 10 J=1,2
      DO 10 I=1,10
10    UNTPTR(I,J)=0

      IF(CLTYPE.EQ.4.OR.CLTYPE.EQ.5)CALL LDUMSG
20    CALL SELUNT(UPTR)
      IF(DRTURN)GOTO70
      CALL CLFMSG(OFF)

C  ****  SEE IF THIS UNIT IS ALREADY IN THE CLUSTER (OMIT IF TRUE)

      DO 30 I=1,UNTMAX
      IF(UNTPTR(I,1).EQ.UPTR)GOTO60
30    CONTINUE
      DO 40 I=1,UNTMAX
      IF(UNTPTR(I,1).EQ.0)GOTO50
40    CONTINUE
      CALL CLFMSG(ON)
      GOTO20

C  ****  ADD UNIT

50    UNTCNT=UNTCNT+1
      UNTPTR(I,1)=UPTR
      CALL GENT(7+I)
      CALL GPUT(1,100,MDF(UPTR+4),0)
      CALL GPUT(2,110,MDF(UPTR+5),0)
      CALL GEON(7+I)
      GOTO20

C  ****  DELETE UNIT

60    UNTCNT=UNTCNT-1
      UNTPTR(I,1)=0
      CALL GEOF(7+I)
      CALL CLFMSG(OFF)
      GOTO20

70    CALL OFFSEL
      CALL BLKSEL

C  ****  PACK THE ARRAY
```

```
      J=1
      DO 75 I=1,15
          IF(UNTPTR(I,1).EQ.0)GOTO75
          UNTPTR(J,1)=UNTPTR(I,1)
          J=J+1
75    CONTINUE

      RETURN
      END
```

```
SUBROUTINE D9
CALL UNTPTH
RETURN
END
```

```
      SUBROUTINE UNTPTH
      COMMON/SINGLE/UPTR,FILPTR
      COMMON/CLUSTR/UNTPTR(10,2),CLTYPE,XZERO,YZERO,XREF,YREF,UNTCNT
      COMMON/OVKLAY/OVLYKY(5)
      IMPLICIT INTEGER (A-Z)

C  ****  CALL ROUTINES WHICH COMPUTE AN INDIVIDUAL UNIT'S PATH IN A CLUSTER

C  ****  'SINGLE UNIT MOVEMENT' PASSES THROUGH THIS ROUTINE

      IF(UPTR.EQ.0)GOTO1
      CALL SNGUNT
      GOTO100

C  ****  SEE WHAT UNIT, IF ANY, NEEDS PROCESSING

1     DO 5 INDEX=1,UNTCNT
         IF(UNTPTR(INDEX,2).EQ.0)GOTO7
5     CONTINUE
      OVLYKY(2)=12
      GO TO 100
7      OVLYKY(2)=OVLYKY(1)+1
      OVLYKY(3)=OVLYKY(1)

C  ****  DETERMINE TYPE OF CLUSTER MOVEMENT

3     BRANCH=CLTYPE-1
      GOTO(10,20,30,40),BRANCH

C  ****  DIRECTION DEPENDENT

10    CALL DIRDEP(INDEX)
      GOTO100

C  ****  FIXED POINT DEPENDENT

20    CALL FIXDEP(INDEX)
```

```
      GOTO100
C   ****  COLUMN
30    CALL COLUMN(INDEX)
      GOTO 100
C   ****  LEAPFROG
40    CALL LPFROG(INDEX)
100   OVLYKY(1)=1
      RETURN
      END
```

```
      SUBROUTINE SNGUNT
      COMMON/SCRTCH/PARRAY(81)
      IMPLICIT INTEGER (A-Z)
      COMMON ARRAY(63)
C  ****  TRANSFERS SINGLE UNIT, USER INPUT PATH, FROM ARRAY TO PARRAY
      DO 10 I=1,63
         PARRAY(I)=ARRAY(I)
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DIRDEP(INDEX)
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/SCRTCH/PARRAY(81)
      COMMON/CLUSTR/UNTPTR(10,2),CLTYPE,XZERO,YZERO,XREF,YREF,UNTCNT
      COMMON ARRAY(63)
      IMPLICIT INTEGER (A-Z)

C  ****  COMPUTES UNIT'S PATH AS MEMBER OF DIRECTION-DEPENDENT CLUSTER

      UPTR=UNTPTR(INDEX,1)

      X=MDF(UPTR+4)
      Y=MDF(UPTR+5)
      PARRAY(1)=ARRAY(1)*3-2
      PARRAY(2)=X
      PARRAY(3)=Y
      XANC=X
      YANC=Y
      X0=ARRAY(2)
      Y0=ARRAY(3)
      X1=ARRAY(4)
```

```
      Y1=ARRAY(5)
C  ****  GET RELATIVE POSITION OF UNIT ON FIRST LEG OF PATH
      CALL RELPOS(X0,Y0,X1,Y1,X,Y,XREL,YREL)

C  ****  GET REAL UNIT POSITION AT END OF FIRST LEG
      X2=2*X1-X0
      Y2=2*Y1-Y0
      CALL REALPS(X1,Y1,X2,Y2,XREL,YREL,X,Y)

C  ****  STUFF THE BUFFER WITH THE UNIT POSITIONS
      PARRAY(4)=X
      PARRAY(5)=Y
C  ****  PROCESS THE REST OF THE CLUSTER PATH LEGS
      XOLD=X
      YOLD=Y
      FLPTR=6
      LIMIT=ARRAY(1)
      DO 10 I=2,LIMIT
         X0=X1
         Y0=Y1
         X1=ARRAY(I*2+2)
         Y1=ARRAY(I*2+3)
         IF(X0.EQ.X1.AND.Y0.EQ.Y1)GOTO5
         CALL REALPS(X0,Y0,X1,Y1,XREL,YREL,X,Y)
C  ****  GET BEST PATH TO THE ROTATION POINT
         CALL PTOL(X,Y,XANC,YANC,XOLD,YOLD,XNEW,YNEW)
         PARRAY(FLPTR-2)=XNEW
         PARRAY(FLPTR-1)=YNEW
5        PARRAY(FLPTR)=X
         PARRAY(FLPTR+1)=Y
C  ****  INSERT WAIT NODE
         PARRAY(FLPTR+2)=X
         PARRAY(FLPTR+3)=Y
         FLPTR=FLPTR+4
         IF(X0.EQ.X1.AND.Y0.EQ.Y1)GOTO10
         X2=2*X1-X0
         Y2=2*Y1-Y0
         CALL REALPS(X1,Y1,X2,Y2,XREL,YREL,X,Y)
         PARRAY(FLPTR)=X
         PARRAY(FLPTR+1)=Y
         FLPTR=FLPTR+2
         XANC=XOLD
         YANC=YOLD
          XOLD=X
          YOLD=Y
10    CONTINUE
      RETURN
```

END

```
      SUBROUTINE FIXDEP(INDEX)
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/SCRTCH/PARRAY(31)
      COMMON/CLUSTR/UNTPTR(10,2),CLTYPE,XZERO,YZERO,XREF,YREF,UNTCNT
      IMPLICIT INTEGER (A-Z)
      COMMON ARRAY(63)

C **** COMPUTES UNIT'S PATH AS MEMBER OF FIXED-POINT-DEPENDENT CLUSTER

      UPTR=UNTPTR(INDEX,1)
      X=MDF(UPTR+4)
      Y=MDF(UPTR+5)
      X0=ARRAY(2)
      Y0=ARRAY(3)

C **** GET RELATIVE POSTION OF UNIT IN THE CLUSTER

      CALL RELPOS(X0,Y0,XREF,YREF,X,Y,XREL,YREL)


      PARRAY(1)=ARRAY(1)*2
      PARRAY(2)=X
      PARRAY(3)=Y

C **** FIND THE REAL POSITION OF THE UNIT ON EACH OF THE LEGS

      FLPTR=4
      LIMIT=ARRAY(1)
      DO 10 I=1,LIMIT
            X0=ARRAY(I*2+2)
            Y0=ARRAY(I*2+3)
            CALL REALPS(X0,Y0,XREF,YREF,XREL,YREL,X,Y)
            PARRAY(FLPTR)=X
            PARRAY(FLPTR+1)=Y
C **** INSERT WAIT NODE
            PARRAY(FLPTR+2)=X
            PARRAY(FLPTR+3)=Y
            FLPTR=FLPTR+4
10    CONTINUE

      RETURN
      END
```

```
SUBROUTINE COLUMN(INDEX)
COMMON/MDFILE/MDF(3000),MDFMAX
COMMON/SCRTCH/PARRAY(81)
```

```
      COMMON/CLUSTR/UNTPTR(10,2),CLTYPE,XZERO,YZERO,XREF,YREF,UCOUNT
       IMPLICIT INTEGER(A-Q,S-Z)
      COMMON ARRAY(63)

C  ****  COMPUTES UNIT'S PATH AS MEMBER OF COLUMN MOVEMENT

      UPTR=UNTPTR(INDEX,1)
      X=MOF(UPTR+4)
      Y=MOF(UPTR+5)
      LIMIT=ARRAY(1)+1

C  ****  PROCESS THE LEAD UNIT

      IF(INDEX.NE.1)GOTO50
      DO 10 I=1,LIMIT
            PARRAY(I*2)=ARRAY(I*2)
            PARRAY(I*2+1)=ARRAY(I*2+1)
10    CONTINUE
      LIMIT=LIMIT-1
      GOTO100

C  ****  PROCESS A FOLLOWING UNIT

50    PARRAY(2)=1
      PARRAY(3)=X
      PARRAY(4)=Y

C  ****  FOR EACH LEAD UNIT MOVEMENT LEG, A FOLLOWING UNIT
C  ****  MOVES AN EQUAL DISTANCE FORWARD ALONG THE SAME PATH

      LEG=1
      NODCNT=1
      LIMIT=LIMIT-1
      DO 90 I=1,LIMIT
           RX=ARRAY(I*2+2)-ARRAY(I*2)
           RY=ARRAY(I*2+3)-ARRAY(I*2+1)
           RLEG=SQRT(RX*RX+RY*RY)
C  ****  ADD UP SEGMENT LENGTHS UNTIL DISTANCE IS REACHED;
C  ****  LABEL EACH NODE ACCORDING TO LEAD UNIT LEG I
70         RX=ARRAY(LEG*2)-X
           RY=ARRAY(LEG*2+1)-Y
           R=SQRT(RX*RX+RY*RY)
           NODCNT=NODCNT+1
           IF (R.GE.RLEG) GO TO 80
                X=ARRAY(LEG*2)
                Y=ARRAY(LEG*2+1)
                LEG=LEG+1
                PARRAY(NODCNT*3-1)=I
                PARRAY(NODCNT*3)=X
                PARRAY(NODCNT*3+1)=Y
                RLEG=RLEG-R
                GO TO 70
80         X=RX*RLEG/R+FLOAT(X)
           Y=RY*RLEG/R+FLOAT(Y)
           PARRAY(NODCNT*3-1)=I+1
           PARRAY(NODCNT*3)=X
```

```
          PARRAY(NODCNT*3+1)=Y
 90    CONTINUE
       PARRAY(NODCNT*3-1)=LIMIT
       LIMIT=NODCNT-1

 100   PARRAY(1)=LIMIT

       RETURN
       END
```

```
      SUBROUTINE LPFROG(INDEX)
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/SCRTCH/PARRAY(61)
      COMMON/CLUSTF/UNTPTR(10,2),CLTYPE,XZERO,YZERO,XREF,YREF,UNTCNT
      IMPLICIT INTEGER(A-Z)
      COMMON ARRAY(63)

C  ****  COMPUTES UNIT'S PATH AS MEMBER OF LEAPFROG MOVEMENT

      PCOUNT=ARRAY(1)*2
      LIMIT=PCOUNT
      UPTR=UNTPTR(INDEX,1)
      X=MDF(UPTR+4)
      Y=MDF(UPTR+5)

      GOTO(10,20),INDEX

C  ****  LEAD UNIT

10    PARRAY(1)=PCOUNT
      FLPTR=2
      DO 15 I=2,LIMIT,4
            PARRAY(FLPTR)=ARRAY(I)
            PARRAY(FLPTR+1)=ARRAY(I+1)
            PARRAY(FLPTR+2)=ARRAY(I)
            PARRAY(FLPTR+3)=ARRAY(I+1)
            PARRAY(FLPTR+4)=ARRAY(I)
            PARRAY(FLPTR+5)=ARRAY(I+1)
            PARRAY(FLPTR+6)=ARRAY(I+2)
            PARRAY(FLPTR+7)=ARRAY(I+3)
            FLPTR=FLPTR+8
15    CONTINUE
      PARRAY(FLPTR)=ARRAY(LIMIT+2)
      PARRAY(FLPTR+1)=ARRAY(LIMIT+3)

      RETURN

C  ****  OTHER UNIT

20    PARRAY(1)=PCOUNT
      PARRAY(2)=X
      PARRAY(3)=Y
      FLPTR=4
```

```
      DO 25 I=2,LIMIT,4
            PARRAY(FLPTR)=ARRAY(I)
            PARRAY(FLPTR+1)=ARRAY(I+1)
            PARRAY(FLPTR+2)=ARRAY(I+2)
            PARRAY(FLPTR+3)=ARRAY(I+3)
            PARRAY(FLPTR+4)=ARRAY(I+2)
            PARRAY(FLPTR+5)=ARRAY(I+3)
            PARRAY(FLPTR+6)=ARRAY(I+2)
            PARRAY(FLPTR+7)=ARRAY(I+3)
            FLPTR=FLPTR+8
25    CONTINUE

      RETURN
      END
```

```
      SUBROUTINE RELPOS(X0,Y0,X1,Y1,X,Y,XREL,YREL)
      IMPLICIT INTEGER (A-Q,S-Z)

C  ****  RETURNS UNIT POSITION RELATIVE TO CLUSTER CENTER
C  ****  AND A REFERENCE POINT
C  ****                     X0,Y0          CLUSTER CENTER
C  ****                     X1,Y1          REFERENCE POINT
C  ****                      X,Y           UNIT POSITION
C  ****                     XREL,YREL     NEW RELATIVE UNIT POSITION

C  ****  TRANSLATE X0,Y0 TO ORIGIN

      RDX=X1-X0
      RDY=Y1-Y0
      XT=X-X0
      YT=Y-Y0

C  ****  ROTATE ABOUT ORIGIN TO Y1=0

      RHYPOT=SQRT(RDX*RDX+RDY*RDY)
      RSIN=RDY/RHYPOT
      RCOS=RDX/RHYPOT

C  ****  GET RELATIVE POSITION OF UNIT IN NEW COORDINATE SYSTEM

      XREL=FLOAT(XT)*RCOS+FLOAT(YT)*RSIN
      YREL=FLOAT(YT)*RCOS-FLOAT(XT)*RSIN

      RETURN
      END
```

```
      SUBROUTINE REALPS(X0,Y0,X1,Y1,XREL,YREL,X,Y)
      IMPLICIT INTEGER(A-Q,S-Z)
C  ****  RETURNS REAL UNIT POSITION WHEN CLUSTER IS AT X0,Y0
```

```
C    ****                           X0,Y0             CLUSTER CENTER
C    ****                           X1,Y1             REFERENCE POINT
C    ****                           XREL,YREL         UNITS RELATIVE POSITION

      RDX=X1-X0
      RDY=Y1-Y0

      RHYPUT=SQRT(RDX*RDX+RDY*RDY)

      RSIN=RDY/RHYPUT
      RCOS=RDX/RHYPUT

      XT=FLOAT(XREL)*RCOS-FLOAT(YREL)*RSIN
      YT=FLOAT(YREL)*RCOS+FLOAT(XREL)*RSIN

      X=XT+X0
      Y=YT+Y0

      RETURN
      END
```

```
      SUBROUTINE PTOL(IX1,IY1,IX2,IY2,IX3,IY3,IX,IY)
C  RETURNS THE POINT ON A GIVEN LINE SEGMENT CLOSEST TO A GIVEN POINT
C              IX1,IY1:   THE POINT TO BE MEASURED FROM THE LINE
C              IX2,IY2:   FIRST ENDPOINT OF LINE SEGMENT
C              IX3,IY3:   SECOND ENDPOINT OF LINE SEGMENT
C              IX,IY  :   POSITION ON LINE SEGMENT CLOSEST TO POINT
      X1=IX1
      Y1=IY1
      A=IY3-IY2
      B=IX3-IX2
      IF(B .NE. 0.) GO TO 5
      IF(((IY1 .GE. IY3) .AND. (IY1 .GE. IY2)) .OR.
     $   ((IY1 .LE. IY3) .AND. (IY1 .LE. IY2))) GO TO 60
      IX=IX2
      IY=IY1
      GO TO 90
5     IF(A) 20,10,20
10    IF(((IX1 .GE. IX3) .AND. (IX1 .GE. IX2)) .OR.
     $   ((IX1 .LE. IX3) .AND. (IX1 .LE. IX2))) GO TO 60
      IX=IX1
      IY=IY2
      GO TO 90
```

```
20    YMAX=IY2
      XMAX=IX2
      YMIN=IY3
      XMIN=IX3
      GO TO 50

30    YMAX=IY3
      XMAX=IX3
      YMIN=IY2
      XMIN=IX2
      GO TO 50

40    IF(B .NE. 0.) GO TO 50
      IF((Y1 .LE. YMIN) .OR. (Y1 .GE. YMAX)) GO TO 60
      IX=IX2
      IY=IY1
      GO TO 90

C  CHECK IF IX1,IY1 IS WITHIN BOUNDARY OF LINE SEGMENT

50    SLOPE=A/B
      SLOPE1=-B/A
      YLMIN=SLOPE1*(X1-XMIN)+YMIN
      YLMAX=SLOPE1*(X1-XMAX)+YMAX
      IF((Y1 .LE. YLMIN) .OR. (Y1 .GE. YLMAX)) GO TO 60

C  IX1,IY1 IS WITHIN BOUNDARY -- COMPUTE CLOSEST POINT ON LINE

      T=FLOAT(IY2)-SLOPE*FLOAT(IX2)
      SLOPE2=SLOPE*SLOPE
      X=(X1+SLOPE*Y1-SLOPE*T)/(1.+SLOPE2)
      IY=SLOPE*X+T+.5
      IX=X+.5
      GO TO 90

C  IX1,IY1 NOT WITHIN BOUNDARY -- CHOOSE CLOSEST ENDPOINT

60    DX=IX1-IX2
      DY=IY1-IY2
      D2=DX*DX+DY*DY
      DX=IX1-IX3
      DY=IY1-IY3
      D3=DX*DX+DY*DY
      IF (D3.GE.D2) GO TO 80

      IX=IX3
      IY=IY3
      GO TO 90

80    IX=IX2
      IY=IY2

90    RETURN
      END
```

```
SUBROUTINE C1[illegible]
CALL TERPTH
RETURN
END
```

```
      SUBROUTINE TERPTH
      COMMON/MOFILE/MDF(3000),MDFMAX
      COMMON/SCRTCH/PARRAY(81)
      COMMON/SINGLE/UPTR,FILPTR
      COMMON/CLUSTR/UNTPTR(10,2),CLTYPE,XZERO,YZERO,XREF,YREF,UNTCNT
      COMMON/OVRLAY/OVLYKY(5)
      IMPLICIT INTEGER (A-Z)

C  ****  ROUTINE BREAKS UNIT PATH IN PARRAY UP INTO SEGMENTS OF CONTINUOUS
C  ****  TERRAIN SO THAT THEY CONFORM TO THE MDF ACTION RECORD FORMAT

       SKRPTR=MDF(3)
      DO 1 I=SKRPTR,MDFMAX
          MDF(I)=0
1     CONTINUE

C  ****  GET FIRST UNPROCESSED UNIT'S MDF AND SCRATCH POINTER

      IF(UPTR.EQ.0)GOTO9
      FILPTR=SKRPTR
      FLPTR=FILPTR+2
      PTR=UPTR
      GOTO20
9     DO 10 I=1,UNTCNT
          IF(UNTPTR(I,2).EQ.0)GOTO16
10    CONTINUE
      OVLYKY(1)=1
      RETURN
16    UNTPTR(I,2)=SKRPTR
      FLPTR=SKRPTR+2
      PTR=UNTPTR(I,1)
      INDEX=I

C  ****  BREAK SEGMENTS INTO TTLEGS (TERRAIN TYPE LEGS)

20    TTYPE0=MDF(PTR+6)
      MASK=189
      SEGCNT=PARRAY(1)
      DO 50 I=1,SEGCNT
          X0=PARRAY(I*2)
          Y0=PARRAY(I*2+1)
          X1=PARRAY(I*2+2)
          Y1=PARRAY(I*2+3)
          IF (CLTYPE.NE.4 .OR. INDEX.EQ.1) GO TO 25
             X0=PARRAY(I*3)
             Y0=PARRAY(I*3+1)
             X1=PARRAY(I*3+3)
```

```
            Y1=PARRAY(I*3+4)
20       CONTINUE
         CALL TTLEGS(X0,Y0,X1,Y1,TTYPE0,MASK,MDF(FLPTR),MDFMAX-FLPTR)
         FINDS=MDF(FLPTR)
         MDF(FLPTR-1)=X0
         MDF(FLPTR)=Y0
         MDF(FLPTR+2)=TTYPE0
         LEG=I
         IF (CLTYPE.EQ.2) LEG=(I+2)/3
         IF (CLTYPE.EQ.3 .OR. CLTYPE.EQ.5) LEG=(I+1)/2
         IF (CLTYPE.EQ.4 .AND. INDEX.GT.1) LEG=PARRAY(I*3-1)
         DO 30 J=0,FINDS
            MDF(FLPTR+3)=LEG
            FLPTR=FLPTR+5
30       CONTINUE
         TTYPE0=MDF(FLPTR-3)
50    CONTINUE

C  ****  PUT LAST POINT OF PATH INTO LIST

      MDF(FLPTR-1)=X1
      MDF(FLPTR)=Y1
      MDF(FLPTR+2)=TTYPE0
      MDF(FLPTR+3)=LEG
      FLPTR=FLPTR+5
      MDF(FLPTR-1)=X1
      MDF(FLPTR)=Y1
      MDF(FLPTR+2)=TTYPE0

      MDF(3)=FLPTR+3

      VLYKY(1)=1

      RETURN
      END
```

```
SUBROUTINE O11
CALL UNTARL
RETURN
END
```

```
SUBROUTINE UNTAPL
COMMON/MDFILE/MDF(3000),MDFMAX
COMMON/OVRLAY/OVLYKY(5)
COMMON/PROBLM/PTIME
COMMON/SINGLE/UPTF,FILPTR
COMMON/UINFO/RCDSIZ
COMMON/RPLAY/INDEX,INTRVL(4),ENDTIM
IMPLICIT INTEGER (A-Z)
COMMON ARRAY(63)
```

```
      DIMENSION TARRAY(31,2)

C  ****  COMPUTES MINIMUM ARRIVAL TIMES FOR SINGLE UNITS
C  ****  ALLOWS USER TO MODIFY THEM ; INSERTS ACTION RECORD INTO UNIT FILE

      SKRPTR=MDF(3)
      UNTBLK=MDF(2)
      ACTBLK=MDF(UNTBLK++RCDSIZ-1)
      ACTRCD=MDF(UPTR+RCDSIZ-1)
      ACTPTR=ACTRCD+MDF(ACTRCD+2)
C  ****  ACTION CODE FOR MOVE TO X,Y, TERRAIN TYPE CHANGE
      BTCODE=3075
C  ****  MOVE TO X,Y
      BTCOD1=3073
      SIZE=MDF(UPTR+2)
      TYPE=MDF(UPTR+3)
      FLPTR=FILPTR
      TTYPE=MDF(UPTR+6)

C  ****  FIND DISTANCES BETWEEN PATH LEG POINTS

      CALL PNTDST

C  ****  COMPUTE ELAPSED TIME OVER EACH SEGMENT OF EACH LEG
C  ****  PLACE IT IN THE ARRIVAL TIME WORD OF THE SEGMENT CELL

15    SPEED=RUNTSP(SIZE,TYPE,TTYPE)
      IF(SPEED.EQ.0)GOTO99
      DIST=MDF(FLPTR+3)
      ELAPSD=FLOAT(DIST)*60./FLOAT(SPEED)+0.5
      MDF(FLPTR+3)=ELAPSD
      FLPTR=FLPTR+5
      TTYPE=MDF(FLPTR-1)
      IF(FLPTR.LT.SKRPTR)GOTO15

C  ****  CUMULATE THE ELAPSED TIMES TO GET THE MINIMUM ARRIVAL TIMES

      FLPTR=FILPTR+5
      CUMTIM=PTIME
      PNTCNT=ARRAY(1)
      DO 25 I=1,PNTCNT
23          CUMTIM=CUMTIM+MDF(FLPTR+3)
            FLPTR=FLPTR+5
            IF(MDF(FLPTR).EQ.I)GOTO23
            TARRAY(I,1)=CUMTIM
25    CONTINUE

C  ****  DISPLAY THE ARRIVAL TIMES AT EACH NODE ON THE PATH
C  ****  AND ALLOW THE USER TO MODIFY THEM

      CALL TTABLE(TARRAY)

C  ****  SCALE THE ARRIVAL TIMES AND CUMULATE

      FLPTR=FILPTR+5
      MDF(FLPTR-2)=PTIME
```

```
30      LEG=MDF(FLPTR)
        IF (TARRAY(LEG,2).GT.0) GO TO 32
        MDF(FLPTR+3)=TARRAY(LEG,1)
        GO TO 34
32      MDF(FLPTR+3)=100.*FLOAT(MDF(FLPTR+3))/FLOAT(TARRAY(LEG,2))+
       1  FLOAT(MDF(FLPTR-2))
34      CONTINUE
        FLPTR=FLPTR+5
        IF(FLPTR.LT.SKRPTR)GOTO30
        MDF(FLPTR-2)=ENDTIM

C  ****  INSERT THE FILE

        START=MDF(UPTR+2*RCDSIZ-1)
        IF(UPTR+RCDSIZ.LT.ACTBLK) GO TO 134
        START=FILPTR
        IF (MDF(4).GT.0) START=MDF(4)
134     CONTINUE
        NEWLEN=SKRPTR-FILPTR
        OLDLEN=START-ACTPTR
        TOTAL=NEWLEN-OLDLEN
        CALL MDFSHF(TOTAL,START)

35      SKRPTR=MDF(3)
        FLEND=SKRPTR-1
        FLBEG=FILPTR+TOTAL
        PTR=ACTPTR

        DO 40 FLPTR=FLBEG,FLEND
              MDF(PTR)=MDF(FLPTR)
              IF(MOD(FLPTR-FLBEG,5).EQ.0)MDF(PTR)=BTCODE
              PTR=PTR+1
40      CONTINUE

50      CALL MVDONE
        MDF(3)=FLBEG
        GOTO100

C  ****  THIS PATH IS BAD  (INTO IMMOBILE TERRAIN)
99      CALL BADPTH(UPTR,MDF(FLPTR-4),MDF(FLPTR-3))
        OVLYKY(1)=1
        RETURN

100     OVLYKY(1)=0
        RETURN

        END
```

```
SUBROUTINE PNTDST
COMMON/MDFILE/MDF(3000),MDFMAX
COMMON/SINGLE/UPTR,FILPTR
COMMON/CLUSTR/CNTPTR(1),2),CLTYPE,XZERO,YZERO,XREF,YREF,CCOUNT
IMPLICIT INTEGER(A-C,S-Z)
```

```
C   ****  COMPUTES DISTANCES BETWEEN PATH LEG POINTS AND PLACES THEM IN
C   ****  THE DESTINATION POINT ARRIVAL TIME SLOT OF IT'S ACTION VECTOR

       SKRPTR=MDF(3)
       FLBEG=UNTPTR(1,2)
      IF(UPTR.NE.0)FLBEG=FILPTR
      FLEND=SKRPTR-5
       X1=MDF(FLBEG+1)
       Y1=MDF(FLBEG+2)
      MDF(FLBEG+3)=0

       DO 1 FLPTR=FLBEG,FLEND,5
           X0=X1
           Y0=Y1
           X1=MDF(FLPTR+6)
           Y1=MDF(FLPTR+7)
          RDX=FLOAT(X1-X0)
          RDY=FLOAT(Y1-Y0)
          DIST=SQRT(RDX*RDX+RDY*RDY)
          IF (UPTR.NE.0) GO TO 6
          DO 5 INDEX=1,UCOUNT
          IF (UNTPTR(INDEX,2).EQ.FLPTR) MDF(FLPTR+3)=0
5         CONTINUE
6          MDF(FLPTR+8)=DIST
1      CONTINUE

       RETURN
       END
```

```
      FUNCTION RUNTSP(SIZE,TYPE,TCODE)
      IMPLICIT INTEGER(A-Z)
      COMMON/SPEED/SPEEDS(4,9)
      COMMON/SCREEN/SSIZE
      DIMENSION BTCODE(6)
      DATA BTCODE/1,2,4,8,16,32/

C  ****  RECIEVES UNIT SIZE,TYPE,AND COMPOSITE TERRAIN LOCATION 'TCODE'
C  ****  COMPUTES UNIT SPEED IN RASTER UNITS/HOUR PROBLEM TIME
C  ****  DECODES 'TCODE', A BIT CODED WORD, FOR SPEED EFFECTIVE TERRAIN

      ROAD=1
      RIVER=2
      LAKE=3
      CITY=4
      HILL=5
      FOREST=6
      HLFRST=7
      HLROAD=8
      CLEAR=9

C  ****  CHECK FOR:
```

```
      TTYPE=0
      IF(TCODE.EQ.0)GOTO10
      IF(IAND(BTCODE(HILL),TCODE).NE.0)TTYPE=HILL
      IF(IAND(BTCODE(FOREST),TCODE).NE.0)TTYPE=FOREST
      IF(IAND(BTCODE(HILL),TCODE).NE.0.AND.TTYPE.EQ.FOREST)TTYPE=HLFRST
      IF(IAND(BTCODE(LAKE),TCODE).NE.0)TTYPE=LAKE
      IF(IAND(BTCODE(CITY),TCODE).NE.0)TTYPE=CITY
      IF(IAND(BTCODE(RIVER),TCODE).NE.0)TTYPE=RIVER
      IF(IAND(BTCODE(ROAD),TCODE).NE.0)TTYPE=ROAD
      IF(TTYPE.EQ.ROAD.AND.IAND(BTCODE(HILL),ICODE).NE.0)TTYPE=HLROAD
10    IF(TTYPE.EQ.0)TTYPE=CLEAR

C  ****  UNIT-TYPE-IN-TERRAIN SPEED

      TSPEED=SPEEDS(TYPE,TTYPE)

C  ****  UNIT SIZE FACTOR (PERCENTAGE)

      SZFCTR=100
      IF(SIZE.EQ.2)SZFCTR=80
      IF(SIZE.EQ.3)SZFCTR=64

C  ****  CONVERT SPEED TO ACCOMODATE UNIT SIZE AND GRAPHICS
C  ****  FUNTSP IS IN GRAPHIC POINTS/HOUR PROBLEM TIME

      FUNTSP=(1023./FLOAT(SSIZE)*FLOAT(TSPEED)*FLOAT(SZFCTR))/100.

      RETURN
      END
```

```
      SUBROUTINE TTABLE(TIMES)
      COMMON/PROBLM/PTIME
      IMPLICIT INTEGER(A-Z)
      COMMON ARRAY(63)
      DIMENSION TIMES(31,2)
C  ****  ROUTINE ALLOWS USER TO SELECT ARRIVAL TIMES AT THE
C  ****  DESTINATIONS OF PATH LEGS WITH THE RESTRICTION THAT THEY
C  ****  NOT PRECEDE THE MINIMUM ARRIVAL TIME IN ARRAY 'TIMES'
      CALL TMSG(TIMES)
      STRTIM=PTIME
      PNTCNT=ARRAY(1)
      IF (PTIME.EQ.TIMES(1,1)) GO TO 200
      DO 100 I=2,PNTCNT
         IF (TIMES(I-1,1).EQ.TIMES(I,1)) GO TO 200
100   CONTINUE
      CALL SELPSP(PRCNTG)
      IF (PRCNTG.LE.0) GO TO 200
      DO 175 I=1,PNTCNT
         MNMUM=TIMES(I,1)
         IF (I.NE.1) STRTIM=TIMES(I-1,1)
         TIME=100.*FLOAT(MNMUM-STRTIM)/FLOAT(PRCNTG)+FLOAT(STRTIM)
```

```
          TIMES(I,2)=RCNTG
          DIFF=TIME-MNMUM
          DO 150 J=1,PNTCNT
             TIMES(J,1)=TIMES(J,1)+DIFF
150       CONTINUE
175    CONTINUE
       CALL TMSG(TIMES)
       RETURN
200    CONTINUE
       DO 10 I=1,PNTCNT
          X=ARRAY(I*2+2)
          Y=ARRAY(I*2+3)
          MNMUM=TIMES(I,1)
          IF(I.NE.1)STRTIM=TIMES(I-1,1)
          NOMNAL=1.25*FLOAT(MNMUM-STRTIM)+FLOAT(STRTIM)

          IF(MNMUM.NE.STRTIM)GOTO2
          CALL SELSTT(MNMUM,X,Y,STRTM1)
          TIME=STRTM1
          TIMES(I,2)=0
          GOTO4
2         CALL SELART(MNMUM,NOMNAL,X,Y,ARTIME)
          TIME=ARTIME
3      TIMES(I,2)=100.*FLOAT(MNMUM-STRTIM)/FLOAT(TIME-STRTIM)
4         DIFF=TIME-MNMUM

          DO 5 J=I,PNTCNT
             TIMES(J,1)=TIMES(J,1)+DIFF
5         CONTINUE
          CALL TMSG(TIMES)
10     CONTINUE

       RETURN
       END
```

```
      SUBROUTINE SELART(MNMUM,NOMNAL,X,Y,TIME)
      COMMON/PROBLM/PTIME
      COMMON/BRANCH/DRTURN
      COMMON/RPLAY/INDEX,INTRVL(4),ENDTIM
      IMPLICIT INTEGER(A-C,E-Z)
      LOGICAL DRTURN

C  ****  ALLOWS USER TO SELECT AN ARRIVAL TIME

      CURSOR=5
      HMTIM1=HRMNS(MNMUM)
      HMTIM2=HRMNS(NOMNAL)

      CALL GENT(CURSOR)
      CALL GPUT(1,100,X,0)
      CALL GPUT(2,110,Y,0)
      CALL GPUT(3,130,3,1)
      CALL GEON(CURSOR)
```

```
1       CALL GHLT

        CALL GSCH(1000,6)
        WRITE(15,1000)
1000    FORMAT('SELECT ARRIVAL TIME')

        CALL GSCH(1001,6)
        WRITE(15,1001)HMTIM1
1001    FORMAT('MINIMUM:',I5,' HOURS')

        CALL GSCH(1002,6)
        WRITE(15,1002)HMTIM2
1002    FORMAT('NOMINAL:',I5,' HOURS')

        CALL GSCH(1003,6)
        WRITE(15,1003)
1003    FORMAT('OTHER')

        CALL GSTT(6,0)

C   ****  SELECT FROM ABOVE LIST
2       CALL SELECT(3,CHOICE)
        IF(DRTURN)GOTO2

        IF(CHOICE.NE.3)GOTO5
3       CALL SELNUM(HMTIM3)
        IF(DRTURN)GOTO1
        CALL DECMIN(HMTIM3,TIME)
        IF(TIME.LT.MNMUM.OR.TIME.GT.ENDTIM)GOTO3
        GOTO 7

5       TIME=MNMUM
        IF(CHOICE.EQ.2)TIME=NOMNAL

7       CALL GENT(CURSOR)
        CALL GPUT(3,130,3,1)
        CALL GECH(CURSOR)

        RETURN
        END
```

```
      SUBROUTINE SELSTT(MNMUM,X,Y,TIME)
      COMMON/RPLAY/INDEX,INTRVL(4),ENDTIM
      IMPLICIT INTEGER(A-Z)

C  ****  ALLOWS USER TO SPECIFY THE DURATION OF A WAIT

      CURSOR=5
      CALL GENT(CURSOR)
      CALL GPUT(1,100,X,0)
      CALL GPUT(2,110,Y,0)
      CALL GPUT(3,130,5,1)
```

```
      CALL GSCN(CURSOR)

1     CALL GHLT

      CALL GSCH(1000,0)
      WRITE(15,1000)
1000  FORMAT('ENTER START TIME OF')

      CALL GSCH(1001,0)
      WRITE(15,1001)
1001  FORMAT('NEXT LEG ON PATH')

      CALL GSTI(0,0)

      CALL SELNUM(HMTIM)
      CALL DECMIN(HMTIM,TIME)
      IF(TIME.LT.MNMUM.OR.TIME.GT.ENDTIM)GOTO1

      CALL GENT(CURSOR)
      CALL GPUT(3,1[illegible],3,0)
      CALL GECF(CURSOR)

      RETURN
      END
```

```
      SUBROUTINE BADPTH(UPTR,X,Y)
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/PROGRM/INTRPT
      IMPLICIT INTEGER(A-Z)

C **** NOTIFIES USER THAT A UNIT HAS A PATH INTO IMMOBILE TERRAIN

      CURSOR=5

C **** BLINK UNIT IN QUESTION
      CALL GENT(MDF(UPTR+1))
      CALL GPUT(3,130,3,1)

C **** POSITION CURSOR AT PATH INTERSECTION WITH IMMOBILE TERRAIN
      CALL GENT(CURSOR)
      CALL GPUT(1,100,X,0)
      CALL GPUT(2,110,Y,0)
      CALL GPUT(3,130,3,1)
      CALL GFCN(CURSOR)

C **** DISPLAY MESSAGE
      CALL BPTHMG
      CALL DNSEL

C **** WAIT FOR AN INTERUPT
      CALL DINTRP

C **** STOP UNIT SYMBOL BLINK
```

```
      CALL GENT(MGF(GPTR+1))
      CALL GPUT(3,130,3,0)

C  ****  TURN OFF CURSOR
      CALL GENT(CURSOR)
      CALL GPUT(3,130,3,0)
      CALL GECF(CURSOR)

C  ****  TURN OFF MESSAGE
      CALL OFFSEL
      CALL BLKSEL

C  ****  TURN OFF CLUSTER CIRCLES AND MOVEMENT PATH
      CALL MVDONE

      RETURN
      END
```

```
      SUBROUTINE DINTRP
      COMMON/KEYLIT/ROW1,ROW2,ROW3,ROW4
      COMMON/PROGRM/INTRPT
      IMPLICIT INTEGER(A-Z)

C  ****  WAITS FOR INTERRUPT  (IN PLACE OF DRECTR)

      CALL LAMPS(248,ROW2,ROW3,ROW4)
1     CALL CKINT(KEY)
      DO 10 J=1,4
          IF(KEY.GT.J*8-6.AND.KEY.LT.J*8)GOTO20
10    CONTINUE
      GOTO1

20    INTRPT=KEY

      RETURN
      END
```

```
      SUBROUTINE MVDONE
      IMPLICIT INTEGER(A-Z)
      COMMON ARRAY(63)
C ****  GRAPHICS INVOLVED IN UNIT MOVEMENT ARE INITIALIZED
C ****  TURN OFF CLUSTER CIRCLES
      DO 1 I=1,10
1     CALL GEOF(7+I)
      RETURN
      END
```

```
SUBROUTINE O12
CALL CLSARL
RETURN
END
```

```
      SUBROUTINE CLSARL
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/BRANCH/DRTURN
      IMPLICIT INTEGER (A-Z)
      LOGICAL DRTURN
      DIMENSION LSTIME(31,2),SPDTBL(31)

C  ****  PERFORMS ARRIVAL TIME DEFINITION FOR CLUSTER UNITS

      OVLYKY(1)=8
      CALL PNTDST
      CALL CTMTBL(LSTIME,SPDTBL)
      IF (DRTURN) RETURN
      CALL TTABLE(LSTIME)
      CALL INSERT(LSTIME,SPDTBL)
      RETURN
      END
```

```
      SUBROUTINE CTMTBL(LSTIME,SPDTBL)
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/CLUSTR/UNTPTR(10,2),CLTYPE,XZERO,YZERO,XREF,YREF,UCOUNT
      COMMON/BRANCH/DRTURN
      COMMON/PROBLM/PTIME
      COMMON ARRAY(63)
      IMPLICIT INTEGER(A-Z)
      LOGICAL DRTURN
      DIMENSION LSTIME(31,2),SPDTBL(31)

C  ****  ROUTINE COMPUTES SLOWEST UNIT ON EACH CLUSTER MOVEMENT
C  ****  LEG FOR THE SPEED TABLE, AND COMPUTES MINIMUM ARRIVAL
C  ****  TIMES FOR EACH NODE OF THE CLUSTER PATH IN THE ARRAY LSTIME

      SKFPTR=MDF(3)
      DRTURN=.FALSE.
      DO 10 LEG=1,31
          LSTIME(LEG,1)=0
          LSTIME(LEG,2)=0
          SPDTBL(LEG)=32767
10    CONTINUE

C  ****  COMPUTE SLOWEST SPEEDS AND LONGEST DISTANCES ALONG EACH LEG
```

```
      DO 20 INDEX=1,UCOUNT
          LEG=0
          SIZE=MDF(UNTPTR(INDEX,1)+2)
          TYPE=MDF(UNTPTR(INDEX,1)+3)
C   ****  FIND THE LEG
          FLPTR=UNTPTR(INDEX,2)+5
1         LEG=LEG+1
2         IF(MDF(FLPTR).EQ.LEG)GOTO4
          FLPTR=FLPTR+5
          IF(FLPTR.GE.UNTPTR(INDEX+1,2).OR.FLPTR.GE.SKRPTR)GOTO20
          GOTO2
C   ****  COMPUTE THE MAXIMUM UNIT MOVEMENT DISTANCE FOR THIS LEG
4         TIME=0
6         TCODE=MDF(FLPTR-1)
          SPEED=RUNTSP(SIZE,TYPE,TCODE)
          IF(SPEED.EQ.0)GOTO99
          IF (SPEED.LT.SPDTBL(LEG)) SPDTBL(LEG)=SPEED
          TIME=MDF(FLPTR+3)+TIME
          FLPTR=FLPTR+5
          IF(TIME.GT.LSTIME(LEG,1))LSTIME(LEG,1)=TIME
          IF(MDF(FLPTR).NE.LEG)GOTO1
          GOTO6
20    CONTINUE

C   ****  COMPUTE MINIMUM ARRIVAL TIMES FROM SPEEDS AND DISTANCES,
C   ****  CUMULATE THE ARRIVAL TIMES

      TIME=PTIME
      NLEGS=ARRAY(1)
      DO 40 LEG=1,NLEGS
         TIME=TIME+IFIX(FLOAT(LSTIME(LEG,1))*60./FLOAT(SPDTBL(LEG)))
         LSTIME(LEG,1)=TIME
40    CONTINUE
      GOTO100

C   ****  UNIT SPEED IS 0.  TELL USER. PATH IS REJECTED AUTOMATICALLY.
99    CALL BADPTH(UNTPTR(INDEX,1),MDF(FLPTR-4),MDF(FLPTR-3))
      DRTURN=.TRUE.

100   RETURN
      END
```

```
SUBROUTINE INSERT(LSTIME,SPDTBL)
COMMON/MDFILE/MDF(3000),MDFMAX
COMMON/CLUSTR/UNTPTR(10,2),CLTYPE,XZERO,YZERO,XREF,YREF,UCOUNT
COMMON/UINFO/RCDSIZ
COMMON/RPLAY/IND,INTRVL(4),ENDTIM
COMMON/PROBLM/PTIME
IMPLICIT INTEGER(A-Z)
REAL SPEED
DIMENSION LSTIME(31,2),SPDTBL(31)
```

```
C    ****   THIS ROUTINE COMPUTES UNIT ARRIVAL TIMES CORRESPONDING
C    ****   TO THE CLUSTER INFO IN LSTIME AND SPDTBL, INSERTS THIS
C    ****   INTO EACH UNIT'S FILE, AND THEN INSERTS EACH FILE INTO
C    ****   THE UNIT'S ACTION RECORD

          SKRPTR=MDF(3)
          DO 20 INDEX=1,UCCUNT
             LEG=0
             TIME=PTIME
C    ****   FIND THE LEG
             FLPTR=UNTPTR(INDEX,2)+5
1            LEG=LEG+1
2            IF(MDF(FLPTR).EQ.LEG)GOTO4
             FLPTR=FLPTR+5
             IF(FLPTR.GE.UNTPTR(INDEX+1,2).OR.FLPTR.GE.SKRPTR)GOTO10
             GOTO2
C    ****   COMPUTE THE ARRIVAL TIMES FOR CELLS ON THE LEG
4            IF (LSTIME(LEG,2).GT.0) GO TO 6
C    ****   PROCESS A WAIT LEG
5            TIME=LSTIME(LEG,1)
             MDF(FLPTR+3)=TIME
             FLPTR=FLPTR+5
             IF (MDF(FLPTR).NE.LEG) GO TO 1
             GO TO 5

6            SPEED=0.01*FLOAT(LSTIME(LEG,2))*FLOAT(SPDTBL(LEG))
7            DIST=MDF(FLPTR+3)
             IF (DIST.GT.0) GO TO 8
C    ****   WAIT FOR OTHER UNITS TO CATCH UP
             TIME=LSTIME(LEG,1)
             MDF(FLPTR+3)=TIME
             GO TO 9
8            TIME=60.*FLOAT(DIST)/SPEED+FLOAT(TIME)
             IF (TIME.GT.LSTIME(LEG,1)) TIME=LSTIME(LEG,1)
             MDF(FLPTR+3)=TIME
9            FLPTR=FLPTR+5
             IF(MDF(FLPTR).NE.LEG)GOTO1
             GO TO 7
10           MDF(FLPTR-7)=ENDTIM
20        CONTINUE

C    ****   INSERT THE FILE

          UNTBLK=MDF(2)
          ACTBLK=MDF(UNTBLK+4+RCDSIZ-1)
C    ****   ACTION CODE FOR MOVE TO X,Y, TERRAIN TYPE CHANGE
          BTCODE=3075
          DO 45 INDEX=1,UCOUNT
          UPTR=UNTPTR(INDEX,1)
          FILPTR=UNTPTR(INDEX,2)
          ACTRCD=MDF(UPTR+RCDSIZ-1)
          ACTPTR=ACTRCD+MDF(ACTRCD+2)
          START=MDF(UPTR+2*RCDSIZ-1)
          IF(UPTR+RCDSIZ.LT.ACTBLK) GO TO 25
          START=FILPTR
          IF (MDF(4).GT.0) START=MDF(4)
```

```
25    CONTINUE
      NEWLEN=SKRPTR-FILPTR
      IF (INDEX.LT.UCOUNT) NEWLEN=UNTPTR(INDEX+1,2)-FILPTR
      OLDLEN=START-ACTPTR
      TOTAL=NEWLEN-OLDLEN
      CALL MDFSHF(TOTAL,START)
      DO 30 I=1,UCOUNT
         UNTPTR(I,2)=UNTPTR(I,2)+TOTAL
30    CONTINUE
      SKRPTR=MDF(3)
      FLEND=SKRPTR-1
      IF (INDEX.LT.UCOUNT) FLEND=UNTPTR(INDEX+1,2)-1
      FLBEG=FILPTR+TOTAL
      PTR=ACTPTR
      DO 40 FLPTR=FLBEG,FLEND
         MDF(PTR)=MDF(FLPTR)
         IF(MOD(FLPTR-FLBEG,5).EQ.0)MDF(PTR)=BTCODE
         PTR=PTR+1
40    CONTINUE
45    CONTINUE
      CALL MVDONE
      MDF(3)=UNTPTR(1,2)
      RETURN
      END
```

```
SUBROUTINE O13
CALL AOBE1
RETURN
END
```

```
SUBROUTINE AOBE1
COMMON/SINGLE/UPTR,FILPTR
COMMON/OVRLAY/OVLYKY(5)
INTEGER OVLYKY
INTEGER UPTR,FILPTR
INTEGER UNTPTR

OVLYKY(1)=1
OVLYKY(2)=17
OVLYKY(3)=13
UNTPTR=UPTR
CALL AOBE(UNTPTR)
RETURN
END
```

```
      SUBROUTINE AOBE(UNTPTR)
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/DISPL/IDPL(4900),IERR
      COMMON/ATT/IATT(12)
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/SCREEN/SIZE
      COMMON/SCRTCH/SIDES(81)
      REAL IDFRAD(4)
      INTEGER SIDES
      INTEGER CODE,DOWN,OVLYKY,SIZE,UNTPTR
      INTEGER USIZE,UTYPE,UTNUM
      DIMENSION PRMTX(3,4,2)
      DIMENSION IARY(150),INTARY(5),AOBMAX(4)
      DATA AOBMAX/.5,2.5,1.,1.5/
      DATA IDFRAD/16.,1024.,3.5,3.5/
      DATA PRMTX/0.,0.,.46,3.75,1.18,.48,3.75,1.18,.48,3.75,1.18,.48,
     +           0.,0.,.24,1.87,.59,.24,1.5,.47,.19,1.87,.59,.24/

      CODE=1
      IF(UNTPTR.GE.MDF(MDF(2)+3))CODE=0
      UTYPE=MDF(UNTPTR+3)
      CALL GLMP(1,30,1)
      CALL ONSEL
      CALL GSCH(1000,35)
      WRITE(15,1000)
 1000 FORMAT('AOBE')
      CALL GSCH(1001,6)
      IF(UTYPE.NE.2)WRITE(15,1001)
      IF(UTYPE.EQ.2)WRITE(15,2001)
 1001 FORMAT('DIRECT AND INDIRECT')
 2001 FORMAT('DIRECT FIRE')
      CALL GSCH(1003,6)
      WRITE(15,1003)
 1003 FORMAT('PRESS RETURN TO')
      CALL GSCH(1004,6)
      WRITE(15,1004)
 1004 FORMAT('SELECT NEW UNIT')
      CALL GSAVE(IDPL,I1,I2,I3)

      UTNUM=MDF(UNTPTR+1)
      DOWN=73
      IEL=6
      H=AOBMAX(UTYPE)*1023./FLOAT(SIZE)
      USIZE=MDF(UNTPTR+2)
      MOVE=MDF(UNTPTR+7)+1
      PR=PRMTX(USIZE,UTYPE,MOVE)*1023./FLOAT(SIZE)

      IF(UTYPE.EQ.2)GO TO 120
           CALL GBEG(1310,MDF(UNTPTR+4),MDF(UNTPTR+5))
           CALL GPUT(4,140,5,CODE)
           CALL GPUT(4,140,6,CODE)
           IRAD=IDFRAD(UTYPE)*1023./FLOAT(SIZE)+PR
           CALL GPUT(5,1760,0,0)
           CALL GPUT(6,1600,IRAD,0)
  120 CONTINUE
      X=MDF(UNTPTR+4)
      Y=MDF(UNTPTR+5)
```

```
      CALL GREG(1320,MDF(UNTPTR+4),MDF(UNTPTR+5))
      CALL GPUT(5,1760,0,0)
      CALL GPUT(6,150,2,3)
      CENTER=IAND(MDF(UNTPTR+6),16)

C     SELECT COLOR FOR THE DETECTION CONTOUR
      IF(UNTPTR.GE.MDF(MDF(2)+3))CODE=0
      CALL GPUT(4,140,5,CODE)
      CALL GPUT(4,140,6,CODE)

      LOOPND=72
      IF(USIZE.EQ.3)LOOPND=36

      DO 250 I=0,LOOPND
      ANGLE=FLOAT(I)*6.28319/FLOAT(LOOPND)
           S=SIN(ANGLE)
           C=COS(ANGLE)
           X1=PR*C+X
           Y1=PR*S+Y
           IX1=X1+.5
           IY1=Y1+.5

C     THE FOLLOWING CODE FINDS THE ENDPOINT X2,Y2 WHICH IS THE FURTHEST
C     POSSIBLE THE GIVEN UNIT MAY SEE ALONG A LINE SEGMENT

           IF(X1.GE.0.AND.X1.LT.1023.AND.Y1.GE.0.AND.Y1.LE.1023.)
     +     GO TO 209
           IX2=X1
           IY2=Y1
           GO TO 214

  209      X2=X1+H*C
           Y2=Y1+H*S

           IF(X2.GE.0.) GO TO 210
                X2=0.
                Y2=Y1+(X2-X1)*S/C
  210      IF(X2.LE.1023.) GO TO 211
                X2=1023.
                Y2=Y1+(X2-X1)*S/C
  211      IF(Y2.GE.0.) GO TO 212
                Y2=0.
                X2=X1+(Y2-Y1)*C/S
  212      IF(Y2.LE.1023.)GO TO 213
                Y2=1023.
                X2=X1+(Y2-Y1)*C/S
  213 CONTINUE
      IX2=X2 + 0.5
      IY2=Y2 + 0.5

      IF (SIDES(1).LE.0) GO TO 430
      IEND=SIDES(1)*8
      DO 420 K=1,IEND,4
      CALL WHEREX(IX1,IY1,IX2,IY2,SIDES(K+1),SIDES(K+2),SIDES(K+3),
     +     SIDES(K+4),INTARY)
```

```
      IF(INTARY(1).EQ.0)GO TO 420
      IX2=INTARY(2)
      IY2=INTARY(3)
  420 CONTINUE


  430      ITYPE=IWRAMI(IX1,IY1,184)
           CALL TTLEGS(IX1,IY1,IX2,IY2,ITYPE,184,IARY,150)
           CALL NOAGGE(IX1,IY1,IX2,IY2,UTYPE,IARY)
  214      IEL=IEL+1
           CALL GPUT(IEL,DOWN,IX2,IY2)
           DOWN=53
  250 CONTINUE

  300 IATT(1)=0
      CALL GSTT(0,0)

  310 CALL GEON(1310)
      CALL GEON(1320)
      CALL GEON(UTNUM)
      CALL DELAY(300)
      IF(IATT(1).NE.0)GO TO 320

      CALL GEOF(1310)
      CALL GEOF(1320)
      CALL GEOF(UTNUM)
      CALL DELAY(60)
  320 IF(IATT(1).EQ.0)GO TO 310
      IF(IATT(1).NE.30.OR.IATT(3).NE.30)GO TO 300

  999 CALL GREST(IDPL,I1,I2,I3)
      CALL GEMP(1,30,0)
      CALL GEON(UTNUM)
      CALL OFFSEL
      CALL BLKSEL
      RETURN
      END
```

```
      SUBROUTINE NDAOBE(IX1,IY1,IX2,IY2,UTYPE,IARY)
C***      THIS SUBROUTINE WILL RETURN IN THE VARIABLE X2 AND Y2 THE COORDINATES
C***      OF THE POINT WHICH INDICATES THE END OF THE LINE OF  THE
C---      BATTLEFIELD EFFECTIVENESS OF A UNIT ALONG  THE LINE SEGMENT
C---      IX1,IY1  IX2,IY2

      COMMON/SCREEN/SIZE
      REAL LOSMTX,LEN1,LEN2
      INTEGER SIZE
      INTEGER TTYP,UTYPE
      INTEGER CT,NT,PNTR,IARY
      DIMENSION IARY(150)
      DIMENSION AOBEMX(7,7,4)
      DATA AOBEMX/3*2.5,0.,2.5,0.,2.5,0.,.5,7*0.,.2,3*.0,.2,3*0.,.5,0.,
```

```
+      .5,0.,2.5,2*.5,4*2.5,5*0.,.2,3*0.,.2,2*0.,2*.2,
+      3*1.5,0.,1.5,0.,1.5,0.,.5,7*0.,.2,3*0.,.2,3*0.,.5,0.,.5,0.,
+      1.5,.5,.5,4*1.5,5*0.,.2,3*0.,.2,2*0.,2*.2,
+      3*1.,0.,1.,0.,1.,0.,.5,7*0.,.2,3*0.,.2,3*0.,.5,0.,.5,0.,
+      1.,2*.5,4*1.,5*0.,.2,3*0.,.2,2*0.,2*.2,
+      3*.5,0.,.5,0.,.5,0.,.5,7*0.,.2,3*0.,.2,3*0.,.5,0.,.5,0.,
+      7*.5,5*0.,.2,0.,2*0.,.2,2*0.,2*.2/
      DIST(A,B,C,D)=SQRT((A-C)*(A-C)+(B-D)*(B-D))

      IHILL=0
      SCALE=1023./FLOAT(SIZE)
      X1=IX1
      Y1=IY1
      X2=IX2
      Y2=IY2
      IARYL=5*IARY(1)+3
      PNTR=5
      IF(IAND(16,IARY(3)).EQ.16)IHILL=-1
      CT=TTYP(IARY(5))
      NT=CT
      LEN1=AOBEMX(CT,CT,OTYPE)*SCALE
      IF(DIST(X1,Y1,X2,Y2).LE.LEN1.AND.IARY(1).EQ.0)RETURN
      IF(IARY(1).EQ.0)GO TO 200
      IF(PNTR.GE.IARYL)RETURN

100   XN=IARY(PNTR)
      YN=IARY(PNTR+1)
      NT=TTYP(IARY(PNTR+3))
      IF((CT.GE.5.AND.CT.NE.6).AND.(NT.EQ.6.OR.NT.LT.5))IHILL=IHILL+1
      IF(DIST(X1,Y1,XN,YN).GE.LEN1)GO TO 200
      LEN2=AOBEMX(CT,NT,OTYPE)*SCALE
      IF((LEN2.EQ.0.).OR.(DIST(X1,Y1,XN,YN).GE.LEN2))GO TO 300
      IF(IHILL.GE.1)GO TO 300
      LEN1=LEN2
      CT=NT
      PNTR=PNTR + 5
      IF(PNTR.LT.IARYL)GO TO 100

200   H=DIST(X1,Y1,X2,Y2)
      RATIO=LEN1/H
      IF(RATIO.GT.1.)RETURN
      IX2=X1+RATIO*(X2-X1)+.5
      IY2=Y1+RATIO*(Y2-Y1)+.5
      RETURN

300   IX2=XN
      IY2=YN
      RETURN
      END
```

```
SUBROUTINE 014
CALL RNGCNT
RETURN
END
```

```
      SUBROUTINE RNGCNT
      COMMON/OVRLAY/OVLYKY(5)
      IMPLICIT INTEGER (A-Z)
C  ****  PERFORMS ALL MMI FOR THE UNIT RANGE CONTOURS
      SIGN=OVLYKY(2)-OVLYKY(1)-1
      IF(SIGN)30,10,20
10    CALL RCMMI1
      GOTO50
20    CALL RCMMI2
      GOTO50
30    CALL RCMMI3
50    OVLYKY(1)=1
      RETURN
      END
```

```
      SUBROUTINE RCMMI1
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/DISPL/IDPL(4900),IERR
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/BRANCH/DRTURN
      IMPLICIT INTEGER(A-C,E-Z)
      LOGICAL DRTURN
      DIMENSION UNTENT(3,4)
      DATA UNTENT/200,201,202,203,204,205,206,207,208,209,210,211/

C  ****  ROUTINE ENABLES USER TO SELECT UNIT POSITIONS,TYPES,SIZES,
C  ****  AND DESTINATION, USED TO COMPUTE RANGE CONTOURS

      ENT=450
      CURSOR=1
      IBALL=1
      ACCEPT=1
      REJECT=0
      SKRPTR=MDF(3)
      FLPTR=SKRPTR+3
      COUNT=0

      CALL GSAVE(IDPL,MDF(2998),MDF(2999),MDF(3000))
```

```
C     **** CREATE RANGE CONTOUR ENTITIES

      DO  35 J=1,10
             CALL GBEG(400+J,0,0)
             CALL COLOR(2)
             CALL GPUT(3,130,2,3)
             CALL GPUT(5,1760,0,1)
35    CONTINUE

1     CALL RCMSG
      IF(COUNT.EQ.10)GOTO5
      CALL SELUTT(TYPE)
      IF(DRTURN.AND.COUNT.EQ.0)GOTO6
      IF(DRTURN)GOTO5

2     IF (TYPE.NE.1) CALL SELUTS(SIZE)
      IF (TYPE.EQ.1) SIZE=3
      IF(DRTURN)GOTO1

      CALL GCPY(ENT,CNTENT(SIZE,TYPE),0,0)
      CALL COLOR(2)

      CALL LAMPS(0,0,0,3)
      CALL UPSMSG
      CALL UNSEL

3     CALL MOVENT(ENT,ENT,TBALL,X,Y)
      CALL CKINT(KEY)

      IF(KEY.NE.REJECT)GOTO4
      CALL GEOF(ENT)
      ENT=ENT+1
      GOTO1

4     IF(KEY.NE.ACCEPT)GOTO3
      I=0.02346*FLOAT(X)+1.5
      J=0.02346*FLOAT(Y)+1.5
      X=42.63*FLOAT(I-1)
      Y=42.63*FLOAT(J-1)
      CALL GPUT(1,100,X,0)
      CALL GPUT(2,110,Y,0)
      CALL SELSPD(PRCNTG)
      IF(.NOT.DRTURN)GOTO45
      CALL GEOF(ENT)
      ENT=ENT+1
      GOTO1
45    CONTINUE
      IF(SIZE.EQ.3)PRCNTG=FLOAT(PRCNTG)*.64
      IF(SIZE.EQ.2)PRCNTG=FLOAT(PRCNTG)*.80

      MOF(FLPTR)=TYPE
      MOF(FLPTR+1)=I
      MOF(FLPTR+2)=J
      MOF(FLPTR+3)=PRCNTG
      FLPTR=FLPTR+4
      COUNT=COUNT+1
```

```
      ENT=ENT+1
      GOTO 1
5     MDF(SKRPTR)=COUNT
      CALL OFFSEL
      CALL BLKSEL
      RETURN
6     DO 7 I=2,5
7     OVLYKY(I)=0
      CALL GREST(IDPL,MDF(2998),MDF(2999),MDF(3000))
      RETURN
      END
```

```
      SUBROUTINE RCMMI2
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/BRANCH/DRTURN
      IMPLICIT INTEGER(A-C,E-Z)
      LOGICAL DRTURN

      SKRPTR=MDF(3)
      ACCEPT=1
      CURSOR=1
      TBALL=1

1     CALL DSTMSG
      CALL RCMSG
      CALL UNSEL
      CALL LAMPS(0,0,0,2)
      CALL GPON(CURSOR)

2     CALL MOVENT(CURSOR,CURSOR,TBALL,X,Y)
      CALL CKINT(KEY)

      IF(KEY.NE.ACCEPT)GOTO2
      I=0.02346*FLOAT(X)+1.5
      J=0.02346*FLOAT(Y)+1.5
      X=42.63*FLOAT(I-1)
      Y=42.63*FLOAT(J-1)
      CALL GPUT(1,100,X,0)
      CALL GPUT(2,110,Y,0)
      MDF(SKRPTR+1)=I
      MDF(SKRPTR+2)=J
      CALL BLKSEL
      CALL CNNMSG
      CALL ONSEL
3     CALL SELNUM(NUMBER)
      IF(DRTURN)GOTO1
      IF(NUMBER.GT.10)GOTO3
      MDF(SKRPTR)=NUMBER
      CALL OFFSEL
      CALL BLKSEL
```

```
CALL GEOF(CURSOR)

RETURN
END
```

```
      SUBROUTINE RCMMI3
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/DISPL/IDPL(4900),IERR
      IMPLICIT INTEGER(A-Q,S-Z)
      INTEGER REJPRS,REJREL,RTURN

C  ****  ALLOWS USER TO VIEW MOVEMENT CONTOURS INDIVIDUALLY
C  ****  TIME LINE AND ARROW REPRESENT RELATIVE PROBLEM TIME
C  ****  NUMERICS ABOVE ARROW INDICATE REAL PROBLEM TIME OF CONTOUR

      REJPRS=100
      REJREL=0
      RTURN=30
      SKRPTR=MDF(3)
      CNTNUM=MDF(SKRPTR)
      ENDTIM=MDF(SKRPTR+1)
      I=CNTNUM
      RNTRVL=FLOAT(ENDTIM)/FLOAT(I)
      INTRVL=1000/I
      CALL RCMSG
      CALL RC2MSG
      CALL CNSEL
      CALL TPNBRT(0)

      CALL LAMPS(64,0,0,1)

10    CALL CKINT(KEY)

      IF(KEY.NE.REJPRS)GOTO20
101   CONTINUE
      CALL DELAY(INTRVL)
      I=I+1
      IF(I.NE.CNTNUM+1)GOTO12
      DO 11 K=1,10
11    CALL GEON(400+K)
      GOTO14
12    DO 13 K=1,10
13    CALL GEOF(400+K)
      IF(I.GT.CNTNUM)I=1
      CALL GEON(400+I)
      CTIME=FLOAT(I)*RNTRVL+0.5
      CALL CLOCK2(CTIME,ENDTIM)
14    CALL CKINT(KEY)
      IF(KEY.NE.REJREL)GOTO101

20    IF(KEY.NE.RTURN)GOTO10
      CALL OFFSEL
      CALL BLKSEL
```

```
      CALL CLKUPD
      CALL GREST(IDPL,MDF(2998),MDF(2999),MDF(3000))
      DO 21 K=1,10
21    CALL GEON(400+K)
      CALL TRNBRT(2)

      RETURN
      END
```

```
      SUBROUTINE [illegible](REL TIM,PREENC)
      COMMON/PROBLEM/TIME
      IMPLICIT INTEGER(A-Z)
      [illegible]
      [illegible]

C  ****  UPDATES THE TIME RECORD AND NUMERIC PROBLEM TIME ON SCREE
C  ****  TIME [illegible] SCALED TO "PRESENT" PROBLEM TIME IS TIM+[illegible]
C  ****  AT DISPLAYS, [illegible] NUMERICAL ARROW REPRESENTS JUST RELTIM

      [illegible]
      [illegible]
      DA=A-50
      CALL GPOINT(10,DA,[illegible])

C  *** [illegible]
      [illegible]
      [illegible]

      DO [illegible] J=1,4
         NUMBAL=NR[illegible]/PLACE*PLACE
         [illegible]
         [illegible]
         [illegible]
         PLACE=PLACE*10
      CONTINUE

      [illegible]
      E[illegible]
```

```
SUBROUTINE C15
CALL URISE
RETURN
END
```

```
SUBROUTINE URISE
COMMON/OVRLAY/IOVKEY(5)
```

```
DATA NXGRD/26/,NYGRD/26/
IOVKEY(1)=1
DELX=1023./FLOAT(NXGRD-2)
DELY=1023./FLOAT(NYGRD-2)
CALL RISE(DELX,DELY)
RETURN
END
```

```
      SUBROUTINE RISL(DELX,DELY)
      DIMENSION IARR(100)
      DIMENSION NT(3,2)
      DIMENSION GAMA(27,27),V(27,27),IST(2,125)
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/SCREEN/ISIZE
      DIMENSION IFCB(13)
      DATA IFCB(3),IFCB(8),IFCB(9),IFCB(10)/0,2HVM,2HAT,2HRX/
      DATA V/729*1E10/,IST/250*-1/
      DATA GAMA/729*9999./
      DATA MAX/125/
C   VFAC IS 2**30
      DATA VFAC/Z4FC00000/
      DATA NXP1/27/,NYP1/27/
C
C     START
C
C     IDENTIFY INITIAL SURF POINT
C
      MPTR=MDF(3)
      NUM=MDF(MPTR)
      MPTR=MPTR+3
      NCOUNT=1
      IUNOLD=0
      ISFOLD=0
      CALL RIVRD(IARR)
    1 CONTINUE
      IUNIT=MDF(MPTR)
      IINIT=MDF(MPTR+1)
      JINIT=MDF(MPTR+2)
      ISFACT=MDF(MPTR+3)
      MPTR=MPTR+4
      IF (IUNIT.EQ.IUNOLD .AND. ISFACT.EQ.ISFOLD) GO TO 3
      SFACT=6000.*FLOAT(ISIZE)/FLOAT(ISFACT*(NXP1-3))
      CALL GETG(GAMA,DELX,DELY,IUNIT,SFACT)
      CALL GAMMOD(IARR,GAMA,DELX,DELY,IUNIT,SFACT)
C     IPTR POINTS TO LOC, IN IST CURRENTLY BEING PROCESSED
    3 ISURF=1
      IPTR=1
      IINIT=IINIT+1
      JINIT=JINIT+1
      V(IINIT,JINIT)=0.
```

```
      ISX=IINIT
      ISY=JINIT
      IST(1,1)=ISX
      IST(2,1)=ISY
 5    CALL NTABLE(ISX,ISY,NT,V)
      ISELNB=0
C
C     SELECT NEIGHBOR POINT
C
10    ISELNB=ISELNB+1
      IF(ISELNB.GT.8) GO TO 30
      IX=NT(ISELNB,1)
      IY=NT(ISELNB,2)
      IF(IX.EQ.0 .OR. GAMA(IX,IY).GE.1000.) GO TO 10
C
C     HAS THIS POINT BEEN PROCESSED
C
      IF (V(IX,IY).GE.0.0 .AND. V(IX,IY).LT.1E9) GO TO 2000
      IF (NCOUNT.GT.1) V(IX,IY)=V(IX,IY)/VFAC-1.0
C
C     CREATE NEW LOCATION IN SURF TABLE
C
      CALL SRFAD(IX,IY,ISURF,IST)
C
C     CALCULATE FOM OF NEIGHBOR TRANSITING TO SURF PT.
C
 20   VTRY=FOM(ISX,ISY,IX,IY,GAMA)+V(ISX,ISY)
C
C     IS THIS FOM < EXISTING FOM OF NEIGHBOR?
C
      IF(VTRY.GT.V(IX,IY)) GO TO 10
C
C     REPLACE NEIGHBORS FOM VALUE AND DECISION
C
      V(IX,IY)=VTRY
C
C     MORE NEIGHBORS TO EVALUATE?
C
      IF(ISELNB.LT.8) GO TO 10
C
C     NO MORE NEIGHBORS TO EVALUATE
C     DELETE SURF POINT FROM SURF TABLE
C
30    CONTINUE
      CALL DLTSRF(IPTR,ISURF,IST,V)
C
C     ISURF TELLS HOW MANY POINTS IN SURF TABLE
C
      IF(ISURF.EQ.0) GO TO 5000
C
C     SELECT SURF POINT WITH LOWEST FOM VALUE
      CALL SCHST(IPTR,IST,ISURF,V)
      ISX=IST(1,IPTR)
      ISY=IST(2,IPTR)
      GO TO 5
2000  CONTINUE
```

```
C
C      DETERMINE FOM VALUE OF NEIGHBORS DESTINATION PT.
C
       VTRY=V(IX,IY)
C
C      SURF FOM .LE. DESTINATION FOM?
       IF(VTRY.LE.V(ISX,ISY)) GO TO 10
       GO TO 20


5000   CONTINUE
       CALL OUT2(V)
       NCOUNT=NCOUNT+1
       IF (NCOUNT.GT.NUM) GO TO 7000
       DO 6000 J=1,NYP1
              L=NYP1-J+1
          DO 6000 I=1,NXP1
              V(I,L)=VFAC*(1.0+V(I,L))
6000   CONTINUE
C   INITIALIZE SURF TABLE
       DO 2 I=1,MAX
              IST(1,I)=-1
              IST(2,I)=-1
 2     CONTINUE
       IONOLD=IONIT
       ISFOLD=ISFACT
       GO TO 1
 7000  CONTINUE
       CALL V$OPEN(18,18,IFCB,0)
       WRITE(18) V
       CALL V$CLOS(18,0)
       RETURN
       END
```

```
      SUBROUTINE RIVRD(IARR)
      COMMON/MDFILE/MDF(3000),MDFMAX
      DIMENSION IARR(1)
C
      NARR=0
C  LOOK FOR RIVERS FIRST
      JTYPE=2
      IEND=MDF(2)-1
 5    IPTR=MDF(1)
      IF (IEND.EQ.IPTR) RETURN
 10   INXT=MDF(IPTR)
      ITYPE=MDF(IPTR+1)
      IF (ITYPE.NE.JTYPE) GO TO 400
      IPTR=IPTR+6
      IMAX=INXT-IPTR-1
      DO 200 I=1,IMAX
             IARR(NARR+I)=MDF(IPTR+I)
 200  CONTINUE
```

```
      NARR=NARR+IMAX+2
      IARR(NARR-1)=2000
      IARR(NARR)=2000
C  FINISHED WITH MDF ?
 400  IF (INXT.EQ.IEND) GO TO 500
      IPTR=INXT
      GO TO 10
 500  IARR(NARR+1)=3000
      IARR(NARR+2)=3000
      IF (JTYPE.EQ.7) RETURN
C  DO ROADS NOW
      JTYPE=7
      NARR=NARR+2
      GO TO 5
      END
```

```
      SUBROUTINE GETG(GAMA,DELX,DELY,IUNIT,SFACT)
      INTEGER SPEEDS
      DIMENSION GAMA(27,27)
      COMMON/SPEED/SPEEDS(4,9)
      DATA NXGRD/26/,NYGRD/26/
C
      DO 100 I=2,NXGRD
         DO 100 J=2,NYGRD
            ICODE=ITEFF(I,J,DELX,DELY)
            IF (ICODE.NE.0) GO TO 30
            GAMA(I,J)=SFACT/FLOAT(SPEEDS(IUNIT,9))
            GO TO 100
 30         ITEST=IAND(ICODE,8)
            IF (ITEST.EQ.0) GO TO 40
            GAMA(I,J)=SFACT/FLOAT(SPEEDS(IUNIT,4))
            GO TO 100
 40         ITEST=IAND(ICODE,4)
            IF (ITEST.EQ.0) GO TO 50
            GAMA(I,J)=9999.
            GO TO 100
 50         ITEST=ICODE/16
            GAMA(I,J)=SFACT/FLOAT(SPEEDS(IUNIT,ITEST+4))
 100  CONTINUE
      RETURN
      END
```

```
FUNCTION ITERR(I,J,DELX,DELY)
      IX=DELX*FLOAT(I-2)
      IF (IX.EQ.0) IX=1
      IF (IX.EQ.1023) IX=1022
      IY=DELY*FLOAT(J-2)
      IF (IY.EQ.0) IY=1
```

```
      IF (IY.EQ.1023) IY=1022
      ITERR=IWRAMI(IX,IY,60)
      RETURN
      END
      SUBROUTINE OUT2(V)
      DIMENSION V(27,27)
      DATA NXP1/27/,NYP1/27/
C
      DO 10 J=1,NYP1
         L=NYP1-J+1
         DO 5 I=1,NXP1
            V(I,L)=-1.0-V(I,L)
            IF (V(I,L).LT.0.0) V(I,L)=1E10
    5    CONTINUE
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE GAMMCD(IARR,GAMA,DELX,DELY,IUNIT,SFACT)
      DIMENSION GAMA(27,27)
      DIMENSION IARR(1)
      DATA NXGRD/26/,NXP1/27/
C
C  RIVERS (IG=1) THEN ROADS (IG=2)
      IG=1
      IPTR=1
      IF (IARR(1).EQ.3000) GO TO 500
      XA=IARR(1)
      YA=IARR(2)
      JPREV=YA/DELY+2.5
      IPTR=3
 100  IF (IARR(IPTR).EQ.2000) GO TO 450
 125  XB=IARR(IPTR)
      YB=IARR(IPTR+1)
C  IFLAG=1 DENOTES FINAL COMPLETION OF A LEG
      IFLAG=0
      IF (XB.EQ.XA) GO TO 200
      SLOPE=(YB-YA)/(XB-XA)
C  JFLAG=1 DENOTES IMMINENT COMPLETION OF A LEG
      JFLAG=0
      JA=YA/DELY+2.5
      JB=YB/DELY+2.5
      JMIN=MIN0(JA,JB)
      JMAX=JA+JB-JMIN
      I=XA/DELX+2.5
 150  X=DELX*FLOAT(I-2)
      IF (ABS(X-XB).LE.0.5*DELX) JFLAG=1
      Y=SLOPE*(X-XA)+YA
      J=Y/DELY+2.5
      IF (J.LT.JMIN) J=JMIN
      IF (J.GT.JMAX) J=JMAX
      JSAVE=J
```

```
      CALL GMOD(GAMA,I,J,IG,DELX,DELY,IUNIT,SFACT)
C  FILL IN J VALUES TO PREVIOUS J (JPREV)
 160  IF (J.GE.JPREV .OR. J.EQ.JMAX) GO TO 170
      J=J+1
      CALL GMOD(GAMA,I,J,IG,DELX,DELY,IUNIT,SFACT)
      GO TO 160
 170  IF (J.LT.JPREV .OR. J.EQ.JMIN) GO TO 180
      J=J-1
      CALL GMOD(GAMA,I,J,IG,DELX,DELY,IUNIT,SFACT)
      GO TO 170
 180  I=I+1
      IF (XB.LT.XA) I=I-2
      IF (I.EQ.1 .OR. I.EQ.NXP1) JFLAG=1
      IF (IFLAG.EQ.1) GO TO 300
      JPREV=JSAVE
      IF (JFLAG.EQ.1) GO TO 400
      GO TO 150
C  VERTICAL LINE OR IDENTICAL POINTS
 200  IF (YB.EQ.YA) GO TO 300
      I=XA/DELX+2.5
      J=YA/DELY+2.5
      JFIN=YB/DELY+2.5
      JPREV=JFIN
 250  CALL GMOD(GAMA,I,J,IG,DELX,DELY,IUNIT,SFACT)
      IF (I.LT.NXGRD) CALL GMOD(GAMA,I,J,IG,DELX,DELY,IUNIT,SFACT)
      J=J+1
      IF (J.GT.JFIN) GO TO 300
      GO TO 250
C  GO TO NEXT LEG
 300  IPTR=IPTR+2
      XA=XB
      YA=YB
      GO TO 100
C  FINISH OFF LEG
 400  I=XB/DELX+2.5
      J=YB/DELY+2.5
      CALL GMOD(GAMA,I,J,IG,DELX,DELY,IUNIT,SFACT)
      IFLAG=1
      GO TO 160
C  GO TO NEXT RIVER OR ROAD
 450  IFLAG=0
      IPTR=IPTR+2
      IF (IARR(IPTR).EQ.3000) GO TO 500
      XA=IARR(IPTR)
      YA=IARR(IPTR+1)
      JPREV=YA/DELY+2.5
      IPTR=IPTR+2
      GO TO 125
C  FINISHED WITH A TERRAIN TYPE
 500  IF (IG.EQ.2) RETURN
      IG=2
      GO TO 450
      END
```

```
      SUBROUTINE GMCD(GAMA,I,J,IG,DELX,DELY,IUNIT,SFACT)
      INTEGER SPEEDS
      COMMON/SPEED/SPEEDS(4,9)
      DIMENSION GAMA(27,27)
C
      GAMA(I,J)=9999.
      IF (IG.EQ.1) RETURN
      GAMA(I,J)=SFACT/FLOAT(SPEEDS(IUNIT,1))
      ICODE=ITEPR(I,J,DELX,DELY)
      IF (ICODE.EQ.16 OR .ICODE.EQ.48)
     *    GAMA(I,J)=SFACT/FLOAT(SPEEDS(IUNIT,8))
      RETURN
      END
```

```
      SUBROUTINE NTABLE(I,J,NT,V)
      DIMENSION V(27,27)
      DIMENSION NT(8,2)
      DATA NXP1/27/,NYP1/27/
C
C     THIS ROUTINE GENERATES A TABLE OF INDICES
C     INDICATING NEIGHBORS OF THE POINT (I,J)
C
      IP1=I+1
      IM1=I-1
      JM1=J-1
      JP1=J+1
C
      NT(1,1)=I
      NT(5,1)=I
      DO 5 K=2,4
5     NT(K,1)=IP1
      DO 7 K=6,8
7     NT(K,1)=IM1
C
      NT(3,2)=J
      NT(7,2)=J
      DO 10 K=4,6
10    NT(K,2)=JM1
      NT(1,2)=JP1
      NT(2,2)=JP1
      NT(8,2)=JP1
      DO 20 L=1,8
      IF (V(NT(L,1),NT(L,2)).LT.0.0) GO TO 25
      IF(NT(L,1).GT.NXP1 .OR. NT(L,1).LT.1) GO TO 25
      IF(NT(L,2).GT.NYP1 .OR. NT(L,2).LT.1) GO TO 25
      GO TO 20
25    NT(L,1)=0
      NT(L,2)=0
20    CONTINUE
      RETURN
      END
```

```
      FUNCTION FOM(I1,J1,I2,J2,GAMA)
      DIMENSION GAMA(27,27)
      DATA SQT202/.707107/
      IF(I1.EQ.I2 .OR. J1.EQ.J2) GO TO 10
C
C     DIAGONAL TRANSIT
C
      FOM=SQT202*(GAMA(I1,J1)+GAMA(I2,J2))
      RETURN
C
C     HORIZONTAL OR VERTICAL TRANSIT
C
10    FOM=(GAMA(I1,J1)+GAMA(I2,J2))*.5
      RETURN
      END
```

```
      SUBROUTINE DLTSRF(IPTR,ISURF,IST,V)
      DIMENSION V(27,27),IST(2,125)
C
C     DELETES A POINT FROM SURF TABLE BY REPLACING INDICES
C     BY -1 AND DECREMENTS ISURF.  ALSO ADDS 9
C     TO ID TO INDICATE POINT IS IN LAKE
C
      V(IST(1,IPTR),IST(2,IPTR))=-1.0-V(IST(1,IPTR),IST(2,IPTR))
      IST(1,IPTR)=-1
```

```
ISTI(2,IPTR)=-1
ISURF=ISURF-1
RETURN
END
```

```
      SUBROUTINE SCHST(IPTR,IST,ISURF,V)
      REAL MIN
      DIMENSION V(27,27),IST(2,125)
      DATA MAX/125/
C
C     ROUTINE SEARCHES IST FOR LOWEST FOM.
C     ISURF TELLS HOW MANY PTS. IN SURF TABLE
C
      MIN=1E10
      DO 100 I=1,MAX
      IF(IST(1,I).LE.0) GO TO 100
      IF(V(IST(1,I),IST(2,I)).GE.MIN) GO TO 100
      IPTR=I
      MIN=V(IST(1,I),IST(2,I))
100   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE OUT2(V)
      DIMENSION V(27,27)
      DATA NXP1/27/,NYP1/27/
C
      DO 10 J=1,NYP1
         L=NYP1-J+1
         DO 5 I=1,NXP1
            V(I,L)=-1.0-V(I,L)
            IF(V(I,L).LT.0.0)V(I,L)=1E10
5        CONTINUE
10    CONTINUE
      RETURN
      END
```

```
SUBROUTINE O16
CALL CONGEN
RETURN
END
```

```
      SUBROUTINE CONGEN
      REAL Z(27,27)
      INTEGER WORK(1000)
      EXTERNAL DRAW
      COMMON/OVRLAY/IOVKEY(5)
      COMMON/MDFILE/MDF(3000),MDFMAX
      DIMENSION CVAL(10)
      DIMENSION IFCB(13)
      DATA IFCB(3),IFCB(8),IFCB(9),IFCB(10)/0,2HVM,2HAT,2HRX/
      DATA NX/27/,NY/27/
      IOVKEY(1)=1
C  EVALUATE FUNCTION TO BE PLOTTED
      CALL V$OPEN(18,18,IFCB,0)
      READ(18) Z
      CALL V$CLOS(18,0)
      IPTR=MDF(3)
      NF=MDF(IPTR)
      IFIN=MDF(IPTR+1)
      JFIN=MDF(IPTR+2)
      TMAX=Z(IFIN+1,JFIN+1)
      DO 10 I=1,NF
            CVAL(I)=TMAX*FLOAT(I)/FLOAT(NF)
 10   CONTINUE
      MDF(IPTR+1)=TMAX
C  DRAW THE CONTOUR PLOTS
      CALL GCONTR(Z,27,NX,NY,CVAL,NF,1E6,WORK,DRAW)
      RETURN
      END
```

```
SUBROUTINE DRAW(X,Y,IFLAG,CVAL)
INTEGER HRMNS
INTEGER CLAB
DIMENSION CVAL(10),CLAB(10),JEL(10)
DATA NCH/1/,CLAB/10*0/,JEL/10*0/
DATA XL/42.625/,YL/42.625/
DATA IBLANK/1H /
IH=IFLAG/10
IF (IH.EQ.0) GO TO 40
IENT=IH+400
CALL GENT(IENT)
IEL=JEL(IH)
IL=IFLAG-10*IH
IF (IL.EQ.6) GO TO 30
ICODE=53
IF (IL.EQ.2) ICODE=73
IF (IL.EQ.3) ICODE=73
IX=(X-2.0)*XL
IY=(Y-2.0)*YL
CALL GPUT(IEL,ICODE,IX,IY)
IEL=IEL+1
IF (IL.LT.2) GO TO 30
IF (IL.GT.4) GO TO 30
IF (NCH.LT.1) GO TO 30
```

```
      IF (CLAB(IH).EQ.IBLANK) GO TO 30
      IF (CLAB(IH).NE.0) GO TO 10
      CALL GPUT(IEL,114,-30,0)
      ITIME=CVAL(IH)
      ITIME=HRMNS(ITIME)
      CALL GCHA(IENT,IEL+1,0,1,3)
      CALL GHLT
      WRITE(15,900) ITIME
900   FORMAT(I3)
      IEL=IEL+5
      GO TO 20
10    CALL GPUT(IEL,114,-30,0)
      CALL GCHA(IENT,IEL+1,0,0,1)
      CALL GHLT
      WRITE(15,901) CLAB(IH)
901   FORMAT(I1)
      IEL=IEL+3
20    CALL GSTT(0,0)
      CALL GPUT(IEL,1760,0,1)
      CALL GPUT(IEL+1,73,IX,IY)
      IEL=IEL+2
30    JEL(IH)=IEL
40    RETURN
      END
```

```
C   CONTOUR DRAWING (GCONTR) -- ACM ALGORITHM 531
C   SEE ACM LISTING FOR EXPLANATION AND COMMENTS
      SUBROUTINE GCONTR(Z,NRZ,NX,NY,CV,NCV,ZMAX,BITMAP,DRAW)
      REAL Z(NRZ,1),CV(1)
      INTEGER BITMAP(1)
      INTEGER L1(4),L2(4),IJ(2)
      INTEGER I1(2),I2(2),I3(6)
      REAL XINT(4)
      REAL XY(2)
      EQUIVALENCE (L2(1),IMAX), (L2(2),JMAX), (L2(3),IMIN),
     *  (L2(4),JMIN)
      EQUIVALENCE (IJ(1),I), (IJ(2),J)
      EQUIVALENCE (XY(1),X), (XY(2),Y)
C
      DATA L1(3)/-1/, L1(4)/-1/
      DATA I1/1,0/, I2/1,-1/, I3/1,0,0,1,1,0/
C
      L1(1)=NX
      L1(2)=NY
      DMAX=ZMAX
      X=1.0
      Y=1.0
      CALL DRAW(X,Y,6,CV)
      ICUR=MAX0(1,MIN0(INT(X),NX))
      JCUR=MAX0(1,MIN0(INT(Y),NY))
      CALL FILL0(BITMAP,2*NX*NY*NCV)
      IBKEY=0
 10   I=ICUR
```

```
      J=JCUR
20    IMAX=I
      IMIN=-I
      JMAX=J
      JMIN=-J
      IDIR=0
30    NXIDIR=IDIR+1
      K=NXIDIR
      IF (NXIDIR.GT.3) NXIDIR=0
40    I=IABS(I)
      J=IABS(J)
      IF (Z(I,J).GT.DMAX) GO TO 140
      L=1
50    IF (IJ(L).GE.L1(L)) GO TO 130
      II=I+I1(L)
      JJ=J+I1(3-L)
      IF (Z(II,JJ).GT.DMAX) GO TO 130
      ASSIGN 100 TO JUMP
60    IX=1
      IF (IJ(3-L).EQ.1) GO TO 80
      II=I-I1(3-L)
      JJ=J-I1(L)
      IF (Z(II,JJ).GT.DMAX) GO TO 70
      II=I+I2(L)
      JJ=J+I2(3-L)
      IF (Z(II,JJ).LT.DMAX) IX=0
70    IF (IJ(3-L).GE.L1(3-L)) GO TO 90
80    II=I+I1(3-L)
      JJ=J+I1(L)
      IF (Z(II,JJ).GT.DMAX) GO TO 90
      IF (Z(I+1,J+1).LT.DMAX) GO TO JUMP, (100, 280)
90    IX=IX+2
      GO TO JUMP, (100, 280)
100   IF (IX.EQ.3) GO TO 130
      IF (IX+IBKEY.EQ.0) GO TO 130
      II=I+I1(L)
      JJ=J+I1(3-L)
      Z1=Z(I,J)
      Z2=Z(II,JJ)
      DO 120 ICV=1,NCV
        IF (IGET(BITMAP,2*(NX*(NY*(ICV-1)+J-1)+I-1)+L).NE.0) GO TO 120
        IF (CV(ICV).LE.AMIN1(Z1,Z2)) GO TO 110
        IF (CV(ICV).LE.AMAX1(Z1,Z2)) GO TO 190
110     CALL MARK1(BITMAP,2*(NX*(NY*(ICV-1)+J-1)+I-1)+L)
120   CONTINUE
130   L=L+1
      IF (L.LE.2) GO TO 50
140   L=MOD(IDIR,2)+1
150   IF (IJ(L).GE.L1(K)) GO TO 170
      IJ(L)=IJ(L)+1
      IF (IJ(L).GT.L2(K)) GO TO 160
      GO TO 40
160   L2(K)=IJ(L)
      IDIR=NXIDIR
      IJ(L)=ISIGN(IJ(L),L1(K))
      GO TO 30
```

```
170   IF (IDIR.EQ.NXIDIR) GO TO 160
      NXIDIR=NXIDIR+1
      IJ(L)=L1(K)
      K=NXIDIR
      L=3-L
      IJ(L)=L2(K)
      IF (NXIDIR.GT.3) NXIDIR=0
      GO TO 150
180   IF (IBKEY.NE.0) RETURN
      IBKEY=1
      GO TO 10
190   IEDGE=L
      CVAL=CV(ICV)
      IF (IX.NE.1) IEDGE=IEDGE+2
      IFLAG=2+IBKEY
      XINT(IEDGE)=(CVAL-Z1)/(Z2-Z1)
200   XY(L)=FLOAT(IJ(L))+XINT(IEDGE)
      XY(3-L)=FLOAT(IJ(3-L))
      CALL MARK1(BITMAP,2*(NX*(NY*(ICV-1)+J-1)+I-1)+L)
      CALL DRAW(X,Y,IFLAG+10*ICV,CV)
      IF (IFLAG.LT.4) GO TO 210
      ICUR=I
      JCUR=J
      GO TO 20
210   NI=1
      IF (IEDGE.LT.3) GO TO 220
      I=I-I3(IEDGE)
      J=J-I3(IEDGE+2)
220   DO 250 K=1,4
            IF (K.EQ.IEDGE) GO TO 250
            II=I+I3(K)
            JJ=J+I3(K+1)
            Z1=Z(II,JJ)
            II=I+I3(K+1)
            JJ=J+I3(K+2)
            Z2=Z(II,JJ)
            IF (CVAL.LE.AMIN1(Z1,Z2)) GO TO 250
            IF (CVAL.GT.AMAX1(Z1,Z2)) GO TO 250
            IF (K.EQ.1) GO TO 230
            IF (K.NE.4) GO TO 240
230         ZZ=Z1
            Z1=Z2
            Z2=ZZ
240         XINT(K)=(CVAL-Z1)/(Z2-Z1)
            NI=NI+1
            KS=K
250   CONTINUE
      IF (NI.EQ.2) GO TO 260
      KS=5-IEDGE
      IF (XINT(3).LT.XINT(1)) GO TO 260
      KS=3-IEDGE
      IF (KS.LE.0) KS=KS+4
260   L=KS
      IFLAG=1
      ASSIGN 280 TO JUMP
      IF (KS.LT.3) GO TO 270
```

```
      I=I+I3(KS)
      J=J+I3(KS+2)
      L=KS-2
270   IF (IGET(BITMAP,2*(NX*(NY*(ICV-1)+J-1)+I-1)+L).EQ.0) GO TO 60
      IFLAG=5
      GO TO 290
280   IF (IX.NE.0) IFLAG=4
290   IEDGE=KS+2
      IF (IEDGE.GT.4) IEDGE=IEDGE-4
      XINT(IEDGE)=XINT(KS)
      GO TO 200
      END
```

```
      SUBROUTINE FILL0(BITMAP,N)
      INTEGER BITMAP(1),N
      DATA NBPW/15/
      LOOP=N/NBPW
      NBLW=MOD(N,NBPW)
      IF (LOOP.EQ.0) GO TO 20
      DO 10 I=1,LOOP
           BITMAP(I)=0
10    CONTINUE
20    IF (NBLW.NE.0) BITMAP(LOOP+1)=MOD(BITMAP(LOOP+1),2**(NBPW-NBLW))
      RETURN
      END
```

```
SUBROUTINE MARK1(BITMAP,N)
INTEGER BITMAP(1),N
DATA NBPW/15/
NWORD=(N-1)/NBPW
NBIT=MOD(N-1,NBPW)
I=2**(NBPW-NBIT-1)
BITMAP(NWORD+1)=BITMAP(NWORD+1)+I*(1-MOD(BITMAP(NWORD+1)/I,2))
RETURN
END
```

```
FUNCTION IGET(BITMAP,N)
INTEGER BITMAP(1),N
DATA NBPW/15/
NWORD=(N-1)/NBPW
NBIT=MOD(N-1,NBPW)
IGET=MOD(BITMAP(NWORD+1)/2**(NBPW-NBIT-1),2)
RETURN
END
```

```
SUBROUTINE 017
CALL DNCTR1
RETURN
END
```

```
      SUBROUTINE DNCTR1
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/BRANCH/DRTURN
      LOGICAL DRTURN
      COMMON/ATT/IATT(12)
      COMMON/SINGLE/UPTR,FILPTR
      INTEGER OVLYKY
      INTEGER UPTR,FILPTR
      INTEGER UNTPTR
      OVLYKY(1)=1
10    IATT(1)=0
      DRTURN=.FALSE.
      IF(IATT(1).NE.0)GO TO 10
      CALL SELUNT(UNTPTR)
      IF(DRTURN)OVLYKY(2)=0
      IF(DRTURN)RETURN
      UPTR=UNTPTR
      CALL HILSDS(UNTPTR)
      RETURN
      END
```

```
      SUBROUTINE HILSDS(UNTPTR)
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/SCRTCH/SIDES(81)
      DIMENSION IARY(150)
      INTEGER OVLYKY
      INTEGER X,Y,X2,Y2,X4,Y4,SIDES
      REAL MINS,MAXS
      INTEGER SIDES,OHPTR,ENDIH,PTR
      INTEGER UNTPTR
      DIST(IA,IB,IC,ID)=SQRT(FLOAT((IA-IC)*(IA-IC)+(IB-ID)*(IB-ID)))
      SLOPE(J1,J2)=FLOAT(Y-J2)/FLOAT(X-J1)

      LEN=150
      X=MDF(UNTPTR+4)
      Y=MDF(UNTPTR+5)
      K=2

      SIDES(1)=0
      OHPTR=1
   50 OHPTR=MDF(OHPTR)
      IF(MDF(OHPTR).EQ.0)GO TO 180
      IF(MDF(OHPTR+1).NE.5)GO TO 50
      IHPTR=MDF(OHPTR)
```

```
      ISQR=IOCT(X,Y,MDF(IHPTR+3),MDF(IHPTR+4),MDF(IHPTR+5),
     +    MDF(IHPTR+6))
      IF(ISQR.EQ.0)GO TO 50

      ENDIH=MDF(IHPTR)
      PTR=IHPTR+7
      IHX=MDF(PTR)
      IHY=MDF(PTR+1)
      IF(IHX.EQ.X)IHX=IHX+1
      IF(IHY.EQ.Y)IHY=IHY+1
      MINS=SLOPE(IHX,IHY)
      MAXS=MINS
      MINSX=IHX
      MINSY=IHY
      MAXSX=IHX
      MAXSY=IHY

   70 PTR=PTR+2
      IF(PTR.GE.ENDIH)GO TO 170
      IHX=MDF(PTR)
      IHY=MDF(PTR+1)
C     THE FOLLOWING ALLEVIATES INFINITE OR HORIZONTAL SLOPES
      IF(IHY.EQ.Y)IHY=IHY+1
      IF(IHX.EQ.X)IHX=IHX+1
      SXY=SLOPE(IHX,IHY)
      IF(ISQR.EQ.4.OR.ISQR.EQ.8)GO TO 75
      IF(SXY.LT.MAXS)GO TO 72
      IF(SXY.EQ.MAXS.AND.DIST(X,Y,IHX,IHY).LT.DIST(X,Y,MAXS,MASY))
     +    GO TO 70
          MAXS=SXY
          MAXSX=IHX
          MAXSY=IHY
          GO TO 70

   72 IF((SXY.GT.MINS).OR.
     +    (SXY.EQ.MINS.AND.DIST(X,Y,IHX,IHY).GT.DIST(X,Y,MINSX,MINSY)))
     +        GO TO 70
          MINS=SXY
          MINSX=IHX
          MINSY=IHY
          GO TO 70

C         FIND LEAST POSITIVE AND LEAST NEGATIVE SLOPE FOR WHEN
C         ISQR=1, OR ISQR=2
   75 IF(SXY.LE.0)GO TO 78
      IF((SXY.GT.MAXS.AND.MAXS.GE.0).OR.
     +    (SXY.EQ.MAXS.AND.DIST(X,Y,IHX,IHY).GE.DIST(X,Y,MAXSX,MAXSY)))
     +    GO TO 70
          MAXS=SXY
          MAXSX=IHX
          MAXSY=IHY
          GO TO 70

   78 IF((SXY.LT.MINS.AND.MINS.LE.0).OR.
     +    (SXY.EQ.MINS.AND.DIST(X,Y,IHX,IHY).GE.DIST(X,Y,MINSX,MINSY)))
     +    GO TO 70
```

```
      MINS=SXY
      MINSX=IHX
      MINSY=IHY
      GO TO 70
C     THE POINTS WHERE THE SIDES OF THE HILL AND THE INNER HILL COINCIDE
C     HAVE BEEN FOUND, NAMELY, MAXSX,MAXSY     AND      MINSX,MINSY
  170 SIDES(1)=SIDES(1)+1
      X2=0
      IF(ISQR.EQ.2.OR.ISQR.EQ.4.OR.ISQR.EQ.9.OR.ISQR.EQ.10)X2=1023
      Y2=32000
      YTEMP=FLOAT(MAXSX-X2)/MAXS+FLOAT(MAXSY)+.5
      IF(YTEMP.LT.32000..AND.YTEMP.GT.-32000.)Y2=YTEMP
      IF(YTEMP.LT.-32000.)Y2=-32000
      CALL TTLEGS(MAXSX,MAXSY,X2,Y2,1,16,IARY,LEN)
      SIDES(K)=MAXSX
      SIDES(K+1)=MAXSY
      SIDES(K+2)=IARY(5)
      SIDES(K+3)=IARY(6)
      K=K+4

      X4=1023
      IF(ISQR.EQ.1.OR.ISQR.EQ.4.OR.ISQR.EQ.9.OR.ISQR.EQ.10)X4=0
      YTEMP=FLOAT(MINSX-X4)/MINS+FLOAT(MINSY)+.5
      Y4=32000
      IF(YTEMP.LT.32000..AND.YTEMP.GT.-32000.)Y4=YTEMP
      IF(YTEMP.LT.-32000.)Y4=-32000
      CALL TTLEGS(MINSX,MINSY,X4,Y4,1,16,IARY,LEN)
      SIDES(K)=MINSX
      SIDES(K+1)=MINSY
      SIDES(K+2)=IARY(5)
      SIDES(K+3)=IARY(6)
      K=K+4
      GO TO 50

  150 SIDES(K)=-100
      RETURN
      END
```

```
SUBROUTINE O18
CALL DNCTR2
RETURN
END
```

```
SUBROUTINE DNCTR2
COMMON/SINGLE/UPTR,FILPTR
COMMON/OVRLAY/OVLYKY(5)
INTEGER OVLYKY
INTEGER UPTR,FILPTR
```

```
INTEGER UNTPTR

OVLYKY(1)=1
OVLYKY(2)=17
OVLYKY(3)=18
UNTPTR=UPTR
CALL DTCNTR(UNTPTR)
RETURN
END
```

```
      SUBROUTINE DTCNTR(UNTPTR)

C     THIS SUBROUTINE CONSTRUCTS DETECTION CONTOURS FOR A GIVEN UNIT BY
C     DETERMINING HOW FAR THE UNIT MAY SEE ALONG 72 LINES, EACH 5 DEGREES
C     APART, AND THEN CONNECTING THE END POINTS

      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/DISPL/IDPL(4900),IERR
      COMMON/ATT/IATT(12)
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/SCREEN/SIZE
      COMMON/SCRTCH/SIDES(81)
      COMMON CENTER
      INTEGER CENTER
      INTEGER SIDES
      INTEGER CODE,ITAN,OVLYKY,SIZE,UNTPTR
      INTEGER USIZE,UTYPE,UTNUM
      DIMENSION DCMTX(3,4,2)
      DIMENSION PRMTX(3,4,2)
      DIMENSION IARY(150),INTARY(5)
      DATA PRMTX/0.,0.,.48,3.75,1.18,.48,3.75,1.18,.48,3.75,1.18,.48,
     +          0.,0.,.24,1.87,.59,.24,1.5,.47,.19,1.87,.59,.24/
      DATA DCMTX/0.,0.,.48,3.75,1.18,.48,3.75,1.18,.48,3.75,1.18,.48,
     +          0.,0.,1.07,11.85,2.89,1.07,11.85,2.89,1.07,11.85,2.89,
     +          1.07/

      CALL GLMP(1,30,1)
      CALL UNSEL
      CALL GSCH(1000,35)
      WRITE(15,1000)
 1000 FORMAT('DETECTION CONTOUR')
      CALL GSCH(1001,6)
      WRITE(15,1001)
 1001 FORMAT('PRESS RETURN TO')
      CALL GSCH(1002,6)
      WRITE(15,1002)
 1002 FORMAT('SELECT NEW UNIT')
      CALL GSAVE(IDPL,I1,I2,I3)


      IF(ITSW(1).EQ.0)GO TO 222
      IPTR=MDF(2)+8
      NUMUTS=MDF(MDF(2))+MDF(MDF(2)+1)
```

```
      DO 222 I=1,NUMUTS
         IENTY=1745+1
         LX=MDF(IPTR)
         LY=MDF(IPTR+1)
         USIZE=MDF(IPTR-2)
         UTYPE=MDF(IPTR-1)
         MOVE=MDF(IPTR+3)+1
         IRAD=DCMTX(USIZE,UTYPE,MOVE)*1023./FLOAT(SIZE)+.5
         IF(IRAD.LT.20)GO TO 221
         CALL GBEG(IENTY,LX,LY)
         CALL GPUT(6,1600,IRAD,0)
221      IPTR=IPTR+25
222   CONTINUE

      UTNUM=MDF(UNTPTR+1)
      CODE=1
      DOWN=73
      IEL=6
      H=20.*1023./FLOAT(SIZE)
      USIZE=MDF(UNTPTR+2)
      UTYPE=MDF(UNTPTR+3)
      MOVE=MDF(UNTPTR+7)+1
      PR=PRMTX(USIZE,UTYPE,MOVE)*1023./FLOAT(SIZE)
      X=MDF(UNTPTR+4)
      Y=MDF(UNTPTR+5)
      CALL GBEG(1700,MDF(UNTPTR+4),MDF(UNTPTR+5))
      CALL GPUT(5,1760,0,0)
      CALL GPUT(6,130,2,3)
      CENTER=IAND(MDF(UNTPTR+6),16)

C     SELECT COLOR FOR THE DETECTION CONTOUR
      IF(UNTPTR.GE.MDF(MDF(2)+3))CODE=0
      CALL GPUT(4,140,5,CODE)
      CALL GPUT(4,140,6,CODE)

      DO 250 I=0,72
         ANGLE=.087266*FLOAT(I)+.04
         S=SIN(ANGLE)
         C=COS(ANGLE)
         X1=PR*C+X
         Y1=PR*S+Y
         IX1=X1+.5
         IY1=Y1+.5

C     THE FOLLOWING CODE FINDS THE ENDPOINT X2,Y2 WHICH IS THE FURTHEST
C     POSSIBLE THE GIVEN UNIT MAY SEE ALONG A LINE SEGMENT

         IF(X1.GE.0.AND.X1.LT.1023.AND.Y1.GE.0.AND.Y1.LE.1023.)
     +   GO TO 209
         IX2=X1
         IY2=Y1
         GO TO 214

209      X2=X1+H*C
```

```
          Y2=Y1+H*S

          IF(X2.GE.0.) GO TO 210
              X2=0.
              Y2=Y1+(X2-X1)*S/C
210       IF(X2.LE.1023.) GO TO 211
              X2=1023.
              Y2=Y1+(X2-X1)*S/C
211       IF(Y2.GE.0.) GO TO 212
              Y2=0.
              X2=X1+(Y2-Y1)*C/S
212       IF(Y2.LE.1023.)GO TO 213
              Y2=1023.
              X2=X1+(Y2-Y1)*C/S
213 CONTINUE
      IX2=X2 + 0.5
      IY2=Y2 + 0.5

      IF (SIDES(1).LE.0) GO TO 430
      IEND=SIDES(1)*8
      DO 420 K=1,IEND,8
      CALL WHEREX(IX1,IY1,IX2,IY2,SIDES(K+1),SIDES(K+2),SIDES(K+3),
     +    SIDES(K+4),INTARY)
      IF(INTARY(1).EQ.0)GO TO 410
      IX2=INTARY(2)
      IY2=INTARY(3)

410 CALL WHEREX(IX1,IY1,IX2,IY2,SIDES(K+5),SIDES(K+6),SIDES(K+7),
     +    SIDES(K+8),INTARY)
      IF(INTARY(1).EQ.0)GO TO 420
      IX2=INTARY(2)
      IY2=INTARY(3)

420 CONTINUE

430       ITYPE=IWRAMI(IX1,IY1,184)
          CALL TTLEGS(IX1,IY1,IX2,IY2,ITYPE,184,IARY,150)
          CALL ENDLOS(IX1,IY1,IX2,IY2,IARY)
214       IEL=IEL+1
          CALL GPUT(IEL,DOWN,IX2,IY2)
          DOWN=53
250 CONTINUE

300 CALL GEON(1700)
    CALL GEON(UTNUM)
    CALL TIME(MN1,ML1)
302 IATT(1)=0
    CALL GSTT(0,0)

303 CALL TIME(MN2,ML2)
    LSP=(MN2-MN1)*12000+(ML2-ML1)
    IF(LSP.GT.300)GO TO 310
    DO 304 K=1,4000
304 CONTINUE
    IF(IATT(1).EQ.0)GO TO 303
    IF(IATT(1).EQ.36.AND.IATT(3).EQ.30)GO TO 999
```

```
310 CALL TIME(MN1,ML1)
    CALL GEOF(1700)
    CALL GEOF(UTNUM)
311 IATT(1)=0
    CALL GSTT(0,0)

312 CALL TIME(MN2,ML2)
    LSP=(MN2-MN1)*12000+(ML2-ML1)
    IF(LSP.GT.40)GO TO 300
    DO 315 K=1,4000
315 CONTINUE
    IF(IATT(1).EQ.0)GO TO 312
    IF(IATT(1).NE.36.OR.IATT(3).NE.30)GO TO 311

999 CALL GREST(IDPL,I1,I2,I3)
    CALL GLMP(1,30,0)
    CALL GEON(UTNUM)
    CALL OFFSEL
    CALL BLKSEL
    RETURN
    END
```

```
      SUBROUTINE ENDLOS(IX1,IY1,IX2,IY2,IARY)
C***       THIS SUBROUTINE WILL RETURN IN THE VARIABLE X2 AND Y2 THE COORDINATES
C***       OF THE POINT WHICH INDICATES THE END OF THE LINE OF SIGHT ALONG THER
C***       LINE SEGMENT INDICATED IN THE PARAMETERS

      COMMON/SCREEN/SIZE
      COMMON CENTER
      INTEGER CENTER
      REAL LOSMTX,LEN1,LEN2
      INTEGER SIZE
      INTEGER TTYP
      INTEGER CT,NT,PNTR,IARY
      DIMENSION LOSMTX(7,7)
      DIMENSION IARY(150)
      DATA LOSMTX/8.,0.,8.,0.,20.,0.,20.,0.,.5,0.,0.,0.,0.,0.,0.,0.,
     +     .5,0.,0.,0.,.5,0.,0.,0.,.75,0.,.75,0.,0.,.5,1.,0.,4.,0.,4.,
     +     0.,0.,0.,0.,0.,.25,0.,0.,0.,.5,0.,0.,.4,.4/
      DIST(A,B,C,D)=SQRT((A-C)*(A-C)+(B-D)*(B-D))

      IHILL=0
      SCALE=1023./FLOAT(SIZE)
      X1=IX1
      Y1=IY1
      X2=IX2
      Y2=IY2
      IARYL=5*IARY(1)+3
      PNTR=5
      IF(IAND(16,IARY(3)).EQ.16)IHILL=-1
      CT=TTYP(IARY(3))
      NT=CT
```

```
      LEN1=LOSMTX(CT,CT)*SCALE
      IF(CENTER.EQ.16.AND.CT.EQ.1)LEN1=20*SCALE
      IF(DIST(X1,Y1,X2,Y2).LE.LEN1.AND.IARY(1).EQ.0)RETURN
      IF(IARY(1).EQ.0)GO TO 200
      IF(PNTR.GE.IARYL)RETURN

  100 XN=IARY(PNTR)
      YN=IARY(PNTR+1)
      NT=TTYP(IARY(PNTR+3))
      IF((CT.GE.5.AND.CT.NE.6).AND.(NT.EQ.6.OR.NT.LT.5))IHILL=IHILL+1
      IF(DIST(X1,Y1,XN,YN).GE.LEN1)GO TO 200
      LEN2=LOSMTX(CT,NT)*SCALE
      IF((LEN2.EQ.0.).OR.(DIST(X1,Y1,XN,YN).GE.LEN2))GO TO 300
      IF(IHILL.GE.1)GO TO 300
      IF(IHILL.GE.2)GO TO 300
      LEN1=LEN2
      CT=NT
      PNTR=PNTR + 5
      IF(PNTR.LT.IARYL)GO TO 100

  200 H=DIST(X1,Y1,X2,Y2)
      RATIO=LEN1/H
      IF(RATIO.GT.1.)RETURN
      IX2=X1+RATIO*(X2-X1)+.5
      IY2=Y1+RATIO*(Y2-Y1)+.5
      RETURN

  300 IX2=XN
      IY2=YN
      RETURN
      END
```

```
      SUBROUTINE WHEREX(IA1,IB1,IA2,IB2,IX1,IY1,IX2,IY2,INTARY)
C     THIS SUBROUTINE DETERMINES THE INTERSECTION OF THE LINE SEGMENT
C     IA1,IB1   IA2,IB2 AND IX1,IY1   IX2,IY2.  IF THE TWO LINE SEGMENTS DO NOT
C     CROSS INTARY(1) IS SET TO 0.  IF THE SEGMENTS CROSS AT A POINT THEN
C     INTARY(1) IS SET TO ONE AND THE POINT OF INTERSECTION IS RETURNED IN
C     INTARY(2) AND INTARY(3).    WHEN THE SEGMENTS COINCIDE INTARY(1) IS SET TO
C     2 AND THE SEGMENT OF INTERSECTION IS RETURNED IN INTARY 2-5.
      DIMENSION INTARY(5)
      A1=IA1
      A2=IA2
      B1=IB1
      B2=IB2
      X1=IX1
      X2=IX2
      Y1=IY1
      Y2=IY2
```

```
      IF(A1.LE.A2)GO TO 20
          ATEMP=A1
          A1=A2
          A2=ATEMP
          BTEMP=B1
          B1=B2
          B2=BTEMP

   20 IF(X1.LE.X2)GO TO 30
          XTEMP=X1
          X1=X2
          X2=XTEMP
          YTEMP=Y1
          Y1=Y2
          Y2=YTEMP

   30 IF(A1.GT.X2.OR.X1.GT.A2)GO TO 999
      YMAX=AMAX1(Y1,Y2)
      BMAX=AMAX1(B1,B2)
      YMIN=AMIN1(Y1,Y2)
      BMIN=AMIN1(B1,B2)
      IF(BMAX.LT.YMIN.OR.YMAX.LT.BMIN)GO TO 999

C     SOLVE FOR VERTICAL LINES
      IF(X1.NE.X2)GO TO 200
      IF(A1.NE.A2)GO TO 150
      INTARY(1)=2
      INTARY(2)=IA1
      INTARY(3)=AMAX1(BMIN,YMIN)
      INTARY(4)=IA1
      INTARY(5)=AMIN1(YMAX,BMAX)
      IF(INTARY(3).EQ.INTARY(5))INTARY(1)=1
      RETURN

  150 S1=(B2-B1)/(A2-A1)
      YINT=S1*(X1-A1)+B1+.5
      IF(YINT.GT.BMAX.OR.YINT.LT.BMIN.OR.YINT.GT.YMAX.OR.YINT.LT.YMIN)
     +    GO TO 999
      INTARY(1)=1
      INTARY(2)=IX1
      INTARY(3)=YINT+.5
      RETURN

  200 IF(A1.NE.A2) GO TO 300
      INTARY(1)=1
      INTARY(2)=IA1
      S2=(Y2-Y1)/(X2-X1)
      INTARY(3)=S2*(A1-X1)+Y1+.5
      RETURN

  300 S1=(B2-B1)/(A2-A1)
      S2=(Y2-Y1)/(X2-X1)
      IF(S1.NE.S2)GO TO 500

C     CASES WHERE LINE SEGMENTS HAVE SAME SLOPES
      IF((B1-S2*A1).NE.(Y1-S1*X1))GO TO 999
```

```
      INTARY(1)=0
      INTARY(2)=AMAX1(A1,X1)
      IF(A1.GT.X1)INTARY(3)=B1
      IF(X1.GE.A1)INTARY(3)=Y1
      INTARY(4)=AMIN1(A2,X2)
      IF(A2.LT.X2)INTARY(5)=B2
      IF(X2.LE.A2)INTARY(5)=Y2
      IF((INTARY(2).EQ.INTARY(4)).AND.(INTARY(3).EQ.INTARY(5)))
     +     INTARY(1)=1
      RETURN

  500 XINT=(B1-Y1+S2*X1-S1*A1)/(S2-S1)
      IF(XINT.LT.A1.OR.XINT.GT.A2)GO TO 999
      IF(XINT.LT.X1.OR.XINT.GT.X2)GO TO 999
      INTARY(1)=1
      INTARY(2)=XINT+.5
      INTARY(3)=S1*XINT+B1-S1*A1+.5
      RETURN

  999 INTARY(1)=0
      RETURN
      END
```

```
INTEGER FUNCTION TTYP(CODE)
INTEGER CODE
TTYP=6
IF(CODE.EQ.0)TTYP=1
IF(CODE.EQ.8)TTYP=2
IF(CODE.EQ.16)TTYP=5
IF(CODE.EQ.32)TTYP=3
IF(CODE.EQ.144)TTYP=4
IF(CODE.EQ.48)TTYP=7
RETURN
END
```

```
SUBROUTINE O19
CALL UUDCT
RETURN
END
```

```
      SUBROUTINE UUDCT
C---        THIS ROUTINE FINDS ALL POSSIBLE DETECTIONS AMOUNG THE GREEN
C---        AND RED UNITS.  WHEN A NEW DETECTION OCCURS A LINE IS DRAWN
```

```
C---       BETWEEN THE UNITS.  THE COLOR OF THE LINE CORRESPONDS TO
C---       THE COLOR OF THE DETECTING UNIT.  IF DETECTION IS RECIPROCAL
C---       THE LINE IS YELLOW.  A DETECTED UNIT IS BLINKED UNTIL IT IS
C---       ONCE AGAIN UNDETECTED.LLEVI

      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/ATT/IATT(12)
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/BRANCH/DRTURN
      COMMON/SINGLE/UPR,FILPTR
      COMMON/DISPL/IDPL(4900),IERR
      COMMON/SCREEN/SIZE
      COMMON/SCRTCH/SIDES(81)
      COMMON/PROBLM/TIME
      LOGICAL DCT1,DCT11,DCT2,DCT12
      REAL MAXLOS
      INTEGER CODE
      LOGICAL SAME
      LOGICAL DRTURN,DETECT
      INTEGER SIDES,OVLYKY,TTYP,SIZE
      INTEGER PX1,PY1,DX2,DY2
      INTEGER UNTPT1,UNTPT2,UPTR,FILPTR
      INTEGER UPTRB2,UPR
      INTEGER TIME,GSEE,GNSEE,RSEE,RNSEE
      INTEGER USIZE,UTYPE
      DIMENSION INTARY(5)
      DIMENSION PRMTX(3,4,2),DCMTX(3,4,2),IARY(150)
      DIMENSION IARY1(150)
      DIMENSION RADII(2)
      DATA PRMTX/0.,0.,.48,3.75,1.18,.48,3.75,1.18,.48,3.75,1.18,.48,
     +           0.,0.,.24,1.87,.59,.24,1.5,.47,.19,1.87,.59,.24/
      DATA DCMTX/0.,0.,.48,3.75,1.18,.48,3.75,1.18,.48,3.75,1.18,.48,
     +           0.,0.,1.07,11.85,2.89,1.07,11.85,2.89,1.07,11.85,2.89,
     +           1.07/


      OVLYKY(1)=21
      IF(MDF(4).EQ.0)CALL EVTINT
      IF((MDF(MDF(2)+1).EQ.0).OR.(MDF(MDF(2)).EQ.0))RETURN
   10 IENT=1900
      CALL GSAVE(IDPL,I1,I2,I3)
      UPTR=0
      SAME=.TRUE.
  100 UPTR=UPTR+1
      UPTRB2=2**(UPTR-1)
      GSEE=0
      GNSEE=0
      RSEE=0
      RNSEE=0
      UNTPT1=MDF(2)+4+20*(UPTR-1)
      CALL HILSDS(UNTPT1)
      SCALE=1023./FLOAT(SIZE)
      MAXLOS=20.*SCALE
      IX1=MDF(UNTPT1+4)
      IY1=MDF(UNTPT1+5)
      USIZE=MDF(UNTPT1+2)
```

```
      UTYPE=MDF(UNTPT1+3)
      MOVE=MDF(UNTPT1+7)+1
      PR=PRMTX(USIZE,UTYPE,MOVE)*SCALE
      DR=DCMTX(USIZE,UTYPE,MOVE)*SCALE
      ITYPE=MDF(UNTPT1+6)
      NUMUTS=MDF(MDF(2)+1)
      UNTPT2=MDF(MDF(2)+3)

      DO 800 I=1,NUMUTS
         DCT1=.FALSE.
         DCT2=.FALSE.
         DCT11=.FALSE.
         DCT12=.FALSE.
         IB2=2**(I-1)
         IF(IAND(MDF(UNTPT1+10),IB2).NE.0)DCT11=.TRUE.
         IF(IAND(MDF(UNTPT1+16),IB2).NE.0)DCT12=.TRUE.
         IX2=MDF(UNTPT2+4)
         IY2=MDF(UNTPT2+5)
         RADII(1)=PR
         USIZE=MDF(UNTPT2+2)
         UTYPE=MDF(UNTPT2+3)
         MOVE=MDF(UNTPT2+7)+1
         RADII(2)=DCMTX(USIZE,UTYPE,MOVE)*SCALE
         D=SQRT(FLOAT(IX2-IX1)**2+FLOAT(IY2-IY1)**2)
         IF(D.LT.(RADII(1)+RADII(2)))DCT1=.TRUE.
         IF(DCT1)GO TO 600
         IF(D.GT.(MAXLOS+RADII(1)+RADII(2)))GO TO 700

C---     DETERMINE IF THE LINE SEGMENT IX1,IY1  IX2,IY2 IS BLOCKED
C---     BY A HILLSIDE, AND IF SO THEN BRANCH TO 700
         IF(SIDES(1).LE.0)GO TO 510
         IEND=SIDES(1)*8
         PX1=FLOAT(IX1)+PR/D*FLOAT(IX2-IX1)+.5
         PY1=FLOAT(IY1)+PR/D*FLOAT(IY2-IY1)+.5
         DX2=FLOAT(IX2)+RADII(2)/D*FLOAT(IX1-IX2)+.5
         DY2=FLOAT(IY2)+RADII(2)/D*FLOAT(IY1-IY2)+.5
         DO 505 K=1,IEND,4
            CALL WHEFLX(PX1,PY1,DX2,DY2,SIDES(K+1),SIDES(K+2),
     +      SIDES(K+3),SIDES(K+4),INTARY)
            IF(INTARY(1).NE.0)GO TO 700
  505    CONTINUE

  510    CALL TTLEGS(IX1,IY1,IX2,IY2,ITYPE,184,IARY,150)
         IF(DETECT(IX1,IY1,IX2,IY2,IARY,RADII))DCT1=.TRUE.

  600    RADII(1)=PRMTX(USIZE,UTYPE,MOVE)*SCALE
         RADII(2)=DR
         IF(D.LT.(RADII(1)+RADII(2)))DCT2=.TRUE.
         IF(DCT2)GO TO 700

C---     REVERSE THE ORDER OF IARY IN THE ARRAY IARY1
         N=IARY(1)
         IARY1(1)=N
         IARY1(3)=IARY(3)
         IF(N.EQ.0)GO TO 620
         IARY1(3)=IARY(5*N+3)
```

```
      DO 620 K=1,N
           IARY1(5*K)=IARY(5*(N+1-K))
           IARY1(5*K+1)=IARY(5*(N+1-K)+1)
           IARY1(K*5+3)=IARY(5*(N-K)+3)
  620 CONTINUE
      IF(DETECT(IX2,IY2,IX1,IY1,IARY1,RADII))DCT2=.TRUE.

  700 IF((DCT1.EQ.DCT11).AND.(DCT2.EQ.DCT12))GO TO 790
C---  EITHER A NEW DETECTION OR AN END OF A DETECTION HAS OCCURRED
      IF((DCT1.AND..NOT.DCT11).OR.(DCT2.AND..NOT.DCT12))SAME=.FALSE.
      IF(DCT1.EQ.DCT11)GO TO 750
      IF(.NOT.DCT1)GO TO 720

C---  GREEN(UPTR) DETECTS RED(I)
      GSEE=IOR(GSEE,IB2)
      MDF(UNTPT1+10)=IOR(MDF(UNTPT1+10),IB2)
      MDF(UNTPT2+16)=IOR(MDF(UNTPT2+16),UPTRB2)
      GO TO 750

C---  GREEN(UPTR) NO LONGER SEES RED(I)
  720 MDF(UNTPT1+10)=IEXOR(MDF(UNTPT1+10),IB2)
      MDF(UNTPT2+16)=IEXOR(MDF(UNTPT2+16),UPTRB2)
      GNSEE=IOR(GNSEE,IB2)
      IF(MDF(UNTPT2+16).NE.0)GO TO 750

  750 IF(DCT2.EQ.DCT12)GO TO 780
      IF(.NOT.DCT2)GO TO 760

C---  RED(I) DETECTS GREEN(UPTR)
      MDF(UNTPT2+10)=IOR(MDF(UNTPT2+10),UPTRB2)
      MDF(UNTPT1+16)=IOR(MDF(UNTPT1+16),IB2)
      RSEE=IOR(RSEE,IB2)
      GO TO 780

C---  RED(I) NO LONGER SEES GREEN(UPTR)
  760 MDF(UNTPT2+10)=IEXOR(MDF(UNTPT2+10),UPTRB2)
      MDF(UNTPT1+16)=IEXOR(MDF(UNTPT1+16),IB2)
      RNSEE=IOR(RNSEE,IB2)

  780 IF(.NOT.((DCT1.AND..NOT.DCT11).OR.(DCT2.AND..NOT.DCT12)))
     +GO TO 790
      CODE=3
      IF(DCT2.AND..NOT.DCT12)CODE=0
      IF((DCT1.AND..NOT.DCT11).AND.(DCT2.AND..NOT.DCT12))CODE=1
      CALL GBEG(IENT,IX1,IY1)
      CALL GPUT(5,1760,0,0)
      CALL GPUT(6,130,2,3)
      CALL GPUT(7,53,IX2,IY2)
      CALL COLOR(CODE)
      IENT=IENT+1

  790 UNTPT2=UNTPT2+20
  800 CONTINUE

  825 IF(GSEE.NE.0)CALL STEVNT(TIME,1,UPTR,GSEE)
      IF(GNSEE.NE.0)CALL STEVNT(TIME,-1,UPTR,GNSEE)
```

```
      IF(RSEE.NE.0)CALL STEVNT(TIME,2,UPTR,RSEE)
      IF(RNSEE.NE.0)CALL STEVNT(TIME,-2,UPTR,RNSEE)

  830 IF(UPTR.LT.MDF(MDF(2)))GO TO 100
      CALL BLKOUTS


      IF(IATT(1).NE.36)GO TO 900
      IF(IATT(3).EQ.30)DRTURN=.TRUE.
      IF(DRTURN)GO TO 1000
      IF(IATT(3).EQ.27)FILPTR=-1

  900 IF(SAME)GO TO 1000
      CALL LAMPS(64,0,0,3)
  905 IATT(1)=0
      CALL GSTT(0,0)
  910 CONTINUE
      IF(IATT(1).EQ.0)GO TO 910
      IF(IATT(1).NE.36)GO TO 905
      IF(IATT(3).GT.1.AND.IATT(3).NE.30)GO TO 905

      IF(IATT(3).EQ.30)DRTURN=.TRUE.
 1000 CALL GREST(IDPL,I1,I2,I3)
      IATT(1)=0
      CALL GSTT(0,0)
      CALL LAMPS(72,0,0,0)
      RETURN
      END
```

```
      LOGICAL FUNCTION DETECT(X1,Y1,X2,Y2,IARY,RADII)
C---       THIS ROUTINE FINDS IF A UNIT AT IX1,IY1 WITH A PRESENCE RADIUS
C---       OF RADII(1) CAN SEE A UNIT LOCATED AT IX2,IY2 WITH A DETECTABLE
C---       RADIUS OF RADII(2). LLEVI
      COMMON/SCREEN/SIZE
      COMMON CENTER,ONHIL
      LOGICAL ONHIL
      REAL LOSMTX,LEN1,LEN2
      INTEGER SIZE,TTYP,CT,NT,PNTR,IARY
      INTEGER X1,Y1,X2,Y2,XN,YN,CENTER
      DIMENSION IARY(150),LOSMTX(7,7)
      DIMENSION RADII(2)
      DATA LOSMTX/8.,[illegible].,8.,0.,20.,0.,20.,0.,.5,0.,0.,0.,0.,0.,0.,0.,
     +     .5,0.,0.,0.,.5,0.,0.,0.,.75,0.,.75,0.,8.,.5,1.,0.,4.,0.,4.,
     +     [illegible],.25,[illegible],.5,[illegible],.4,.4/
      DIST(IA,IB,IC,ID)=SQRT((FLOAT(IA-IC))**2+(FLOAT(IB-ID))**2)
      IARYL=5*IARY(1)+3
      SCALE=1023./FLOAT(SIZE)
      PNTR=5
```

```
      CT=TTYP(IARY(3))
      NT=CT
      D=DIST(X1,Y1,X2,Y2)
      CENTER=IAND(16,IARY(3))
      IHILL=0

C          IF NO INTERSECTIONS THEN GO TO 790
      IF(IARY(1).EQ.0)GO TO 790

C          IGNORE INTERSECTIONS WHICH OCCUR WITHIN PRESENCE CIRCLE
   20 XN=IARY(PNTR)
      YN=IARY(PNTR+1)
      NT=TTYP(IARY(PNTR+3))
      IF(DIST(X1,Y1,XN,YN).GT.RADII(1))GO TO 90
      CT=NT
      PNTR=PNTR+5
      IF(PNTR.LT.IARYL)GO TO 20
      GO TO 790

   90 LEN1=LOSMTX(CT,CT)*SCALE+ RADII(1)
      IF(CT.GE.4)IHILL=-1
      IF(CENTER.EQ.16.AND.CT.EQ.1)LEN1=20.*SCALE+RADII(1)
      CENTER=0
      IF(DIST(X1,Y1,XN,YN).GT.LEN1) GO TO 800
      IF(DIST(XN,YN,X2,Y2).LE.RADII(2))GO TO 810
      LEN1=LOSMTX(CT,NT)*SCALE+RADII(1)
      PNTR=PNTR+5
      IF(PNTR.GE.IARYL)GO TO 800

C---       JUMP THRU THE TRANSITION MATRIX
  100 XN=IARY(PNTR)
      YN=IARY(PNTR+1)
      CT=NT
      NT=TTYP(IARY(PNTR+3))
      IF(DIST(X1,Y1,XN,YN).GT.LEN1) GO TO 800
      IF(DIST(XN,YN,X2,Y2).LE.RADII(2))GO TO 810
      IF(CT.GE.4.AND.NT.LE.3)IHILL=IHILL+1
      IF(IHILL.GE.0)GO TO 900
      PNTR=PNTR+5
      LEN1=LOSMTX(CT,NT)*SCALE+RADII(1)
      IF(DIST(X1,Y1,XN,YN).GT.LEN1) GO TO 800
      IF(DIST(XN,YN,X2,Y2).LE.RADII(2))GO TO 810
      GO TO 100

  790 LEN1=LOSMTX(CT,NT)*SCALE+RADII(1)
      IF(CENTER.EQ.16.AND.CT.EQ.1)LEN1=20.*SCALE+RADII(1)
  800 IF(D.GT.(LEN1+RADII(2)))GO TO 900
  810 DETECT=.TRUE.

      RETURN

  900 DETECT=.FALSE.
      RETURN
      END
```

```
      SUBROUTINE STEVNT(TIME,EVENT,UNIT1,UNITS2)
C---       THIS SUBROUTINE STORES EVENTS IN THE EVENT FILE
C---       EVENT=1 DENOTES THAT UNIT1 DETECTS UNITS2
C---       EVENT=2 DENOTS THAT UNIT1 IS DETECTED BY UNITS2
C---       WHEN THE NUMBERS ARE NEGATIVE THE EVENT HAS JUST ENDED
      COMMON/MDFILE/MDF(3000),MDFMAX
      IMPLICIT INTEGER(A-Z)

      EPTR=MDF(MDF(4))
      MDF(EPTR)=TIME
      MDF(EPTR+1)=EVENT
      MDF(EPTR+2)=UNIT1
      MDF(EPTR+3)=UNITS2
      MDF(EPTR+4)=0
      MDF(MDF(4))=EPTR+5
      MDF(3)=EPTR+5
      RETURN
      END
```

```
SUBROUTINE O20
CALL SETIME
RETURN
END
```

```
      SUBROUTINE SETIME
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/PROBLM/PTIME
      COMMON/BRANCH/URTURN
      COMMON/RPLAY/INDEX,INTRVL(4),ENDTIM
      IMPLICIT INTEGER (A-Z)
      LOGICAL URTURN
      LOGICAL EVENTS
C  ****  ALLOWS THE USER TO RESET THE PROBLEM TIME TO AN EARLIER VALUE
      IF(MDF(4).EQ.0)CALL EVTINT
      EPTR=MDF(4)
      EVENTS=.TRUE.
      IF(MDF(3).EQ.(MDF(4)+3))EVENTS=.FALSE.
      OVLYKY(1)=1
      CALL SELTIM(TIME)
      IF (URTURN) RETURN
      IF(TIME.LE.MDF(EPTR+1))GO TO 7
      IF(PTIME.LT.MDF(EPTR+1))TIME=MDF(EPTR+1)
    7 PTIME=
```

```
      NXTIME=ENDTIM+1
      IF(EVENTS)NXTIME=MDF(MDF(4)+3)
      IF(MDF(4).NE.0)MDF(MDF(4))=MDF(4)+3
      CALL UFESET
      CALL DSPLAY
      CALL CLKUPD
      IF(TIME.LE.0)GO TO 25
      CALL BLKUTS
      IF(NXTIME.EQ.0)CALL UPDATE

10    PTIME=PTIME+1
      IF(TIME.GT.MDF(EPTR+1))GO TO 15
      IF(PTIME.LT.NXTIME)GO TO 15
      CALL UPDATE
      NXTIME=ENDTIM+1
      IF(MDF(MDF(4)).LT.MDF(3))NXTIME=MDF(MDF(MDF(4)))
15    CALL RESULT
      CALL MVMENT
      CALL DSPLAY
      CALL CLKUPD
      IF (PTIME.LT.TIME) GO TO 10

      IF(TIME.EQ.MDF(EPTR+1))CALL MESAGE(0)
      IF(TIME.GT.MDF(EPTR+1))CALL MESAGE(1)
25    CALL BLKUTS
      IF(TIME.GE.MDF(EPTR+1))CALL DINTRP
      CALL OFFSEL
      CALL BLKSEL
      RETURN
      END
```

```
      SUBROUTINE URESET
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/UINTL/ENDRNC(3,4),ASSETS(3,4)
      COMMON/UINFO/RCDSIZ
      IMPLICIT INTEGER (A-Z)
C   ****  RESETS UNIT INFO AND ACTION POINTERS TO TIME ZERO
      UPTR=MDF(2)
      UTOTAL=MDF(UPTR)+MDF(UPTR+1)
      UPTR=UPTR+4
      POSTUR=1
      DO 100 J=1,UTOTAL
         ACTPTR=MDF(UPTR+RCDSIZ-1)
         MDF(ACTPTR)=3
         MDF(ACTPTR+1)=3
         MDF(ACTPTR+2)=16
         MDF(UPTR+4)=MDF(ACTPTR+4)
         MDF(UPTR+5)=MDF(ACTPTR+5)
         MDF(UPTR+6)=MDF(ACTPTR+7)
         MDF(UPTR+7)=0
```

```
      SIZE=MDF(UPTR+2)
      TYPE=MDF(UPTR+3)
      MDF(UPTR+8)=ASSETS(SIZE,TYPE)
      MDF(UPTR+9)=ENDRNC(SIZE,TYPE)
      DO 20 K=10,17
         MDF(UPTR+K)=0
20    CONTINUE
      MDF(UPTR+18)=POSTUR
      UPTR=UPTR+RCDSIZ
100   CONTINUE

      RETURN
      END
```

```
      SUBROUTINE UPDATE
C---       THIS SUBROUTINE CHECKS FOR DETECTED AND DETECTING UNITS
C---       AT PTIME AND THEN UPDATES THE UNIT INFORMATION RECORDS
C---       AND THE EVENT POINTER I.E. MDF(MDF(3)). LLEVI

      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/PROBLM/PTIME
      IMPLICIT INTEGER(A-Z)
      REDPTR(N)=MDF(MDF(2)+3)+20*(N-1)

      EVPTR=MDF(MDF(4))
      NUMRED=MDF(MDF(2)+1)

   10 EVENT=MDF(EVPTR+1)
      GRNUNT=MDF(EVPTR+2)
      RDUNTS=MDF(EVPTR+3)
      GRNPTR=MDF(MDF(2)+2)+20*(GRNUNT-1)
      GUB2=2**(GRNUNT-1)

      IF(EVENT.NE.1)GO TO 200
      MDF(GRNPTR+10)=IOR(MDF(GRNPTR+10),RDUNTS)
      UTCODE=1
      DO 150  I=1,NUMRED
           IF(IAND(UTCODE,RDUNTS).EQ.0)GO TO 149
           MDF(REDPTR(I)+16)=IOR(MDF(REDPTR(I)+16),GUB2)
  149      UTCODE=2*UTCODE
  150 CONTINUE
      GO TO 900

  200 IF(EVENT.NE.2)GO TO 400
      MDF(GRNPTR+16)=IOR(MDF(GRNPTR+16),RDUNTS)
      UTCODE=1
      DO 250  I=1,NUMRED
           IF(IAND(UTCODE,RDUNTS).EQ.0)GO TO 249
           MDF(REDPTR(I)+10)=IOR(MDF(REDPTR(I)+10),GUB2)
  249      UTCODE=UTCODE*2
  250 CONTINUE
      GO TO 900
```

```
400 IF(EVENT.NE.-1)GO TO 600
    MDF(GRNPTR+10)=IEXOR(MDF(GRNPTR+10),RDUNTS)
    UTCODE=1
    DO 450 I=1,NUMRED
        IF(IAND(UTCODE,RDUNTS).EQ.0)GO TO 449
        MDF(REDPTR(I)+16)=IEXOR(MDF(REDPTR(I)+16),GUB2)
449     UTCODE=UTCODE*2
450 CONTINUE
    GO TO 900

600 IF(EVENT.NE.-2)GO TO 900
    MDF(GRNPTR+16)=IEXOR(MDF(GRNPTR+16),RDUNTS)
    UTCODE=1
    DO 650  I=1,NUMRED
        IF(IAND(UTCODE,RDUNTS).EQ.0)GO TO 649
        MDF(REDPTR(I)+10)=IEXOR(MDF(REDPTR(I)+10),GUB2)
649     UTCODE=UTCODE*2
650 CONTINUE

900 EVPTR=EVPTR+5
    IF((EVPTR.LT.MDF(3)).AND.(MDF(EVPTR).EQ.PTIME))GO TO 10

    MDF(MDF(4))=EVPTR
    RETURN
    END
```

```
      SUBROUTINE O21
      CALL REPLAY
      RETURN
      END



      SUBROUTINE REPLAY
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/RPLAY/INDEX,INTRVL(4),ENDTIM
      COMMON/PROGRM/INTRPT
      COMMON/PROBLM/PTIME
      COMMON/BRANCH/DRTURN
      COMMON/OVRLAY/OVLYKY(5)
      COMMON/SINGLE/UPTR,FILPTR
      COMMON/OLAY/IOELY
      IMPLICIT INTEGER(A-C,E-Z)
      LOGICAL EVENTS
      LOGICAL DRTURN

C  ****  PERFORMS REPLAY OF PLANNED UNIT ACTIVITIES
C  ****  FINDS DETECTIONS AND FIRING RANGE EVENTS DURING 'CURRENT REPLAY'

      RPSPD=27
```

```
      CURPLY=28
      MSTRPL=29
      RTURN=30
      COUNT=0
      MAXCNT=10
      CALL GHLT
      IF(ITSW(2).AND.ITSW(3))READ(1,700)IDELY
  700 FORMAT(I5)
      CALL GSTT(0,0)
      IF (ITSW(1)) MAXCNT=5

      IF(MDF(4).EQ.0)CALL EVTINT
      EPTR=MDF(4)
      IF((INTRPT.NE.MSTRPL).OR.(PTIME.LT.MDF(EPTR+1)))GO TO 3
      CALL MESAGE(1)
      CALL OCTOUT
      CALL BLKOUTS
    3 NXTIME=ENDTIM+1
      NXTIME=ENDTIM+1
      EVENTS=.TRUE.
      IF(MDF(3).EQ.(MDF(4)+3))EVENTS=.FALSE.
      IF(EVENTS.AND.(MDF(EPTR).LT.MDF(3)))NXTIME=MDF(MDF(EPTR))

5     IF(PTIME.EQ.ENDTIM)INTRPT=-1
      IF(PTIME.EQ.ENDTIM)GOTO75

      CALL CKINT(KEY)

      IF(DRTURN.OR.(KEY.EQ.RTURN))INTRPT=-1
      IF(DRTURN.OR.(KEY.EQ.RTURN))GO TO 75

45    IF(KEY.NE.RPSPD)GOTO50
      CALL RPLSPD
      GOTO10

50    IF (FILPTR.GE.0) GO TO 10
      FILPTR=0
      CALL RPLSPD

   10 IF(INTRPT.NE.MSTRPL)GO TO 15
      IF((PTIME.NE.NXTIME).OR.(PTIME.GT.MDF(EPTR+1)))GO TO 15
      CALL REHASH
      NXTIME=ENDTIM+1
      IF(MDF(EPTR).LT.MDF(3))NXTIME=MDF(MDF(EPTR))
   15 CALL TMEASE

      CALL RESULT

20    CALL MVMENT

30    CALL DSPLAY
      CALL CLKUPD

      IF(INTRPT.EQ.CURPLY)GO TO 60
      IF(PTIME.EQ.MDF(EPTR+1))GO TO 70
      GO TO 5
```

```
50     COUNT=COUNT+1
       MOF(EPTR+1)=PTIME
       IF (COUNT.LT.MAXCNT) GO TO 5
       OVLYKY(1)=19
       RETURN

   70 IF(PTIME.EQ.NXTIME)CALL REHASH
       CALL MESAGE(0)
       CALL DINTRP
75     OVLYKY(1)=1
       CALL OFFSEL
       CALL BLKSEL
       RETURN

       END
```

```
      SUBROUTINE DCTOUT
      COMMON/UINFO/RCDSIZ
      COMMON/MDFILE/MDF(3000),MDFMAX
      IMPLICIT INTEGER(A-Z)
      UPTR=MDF(2)
      UTOTAL=MDF(UPTR)+MDF(UPTR+1)
      IF(UTOTAL.EQ.0)RETURN
      UPTR=UPTR+4
      DO 100 J=1,UTOTAL
         MDF(UPTR+10)=0
         MDF(UPTR+16)=0
         UPTR=UPTR+RCDSIZ
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE REHASH
C---      THIS ROUTINE IS CALLED TO READ AN EVENT FROM THE
C---      EVENT FILE , UPDATE INFORMATION IN THE UNIT INFO RECORDS
C---      AND DRAW DETECTION LINES.  LLEVI
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/PROBLM/PTIME
      COMMON/FPLAY/INDEX,INTRVL(4),ENDTIM
      COMMON/DLAY/IDELY
      IMPLICIT INTEGER(A-Z)
      LOGICAL NEW
      FEDPTR(N)=MDF(MDF(2)+3)+20*(N-1)

      IENT=2140
      NUMRED=MDF(MDF(2)+1)
```

```
      MUTUAL=0
      CALL GSAVE(IDPL,I1,I2,I3)
      NEW=.FALSE.
      EVPTR=MDF(MDF(4))

 10   EVENT=MDF(EVPTR+1)
      GRNUNT=MDF(EVPTR+2)
      GUB2=2**(GRNUNT-1)
      RDUNTS=MDF(EVPTR+3)
      GRNPTR=MDF(MDF(2)+2)+20*(GRNUNT-1)
      EVPTR=EVPTR+5
      MORE=.FALSE.
      IF((EVPTR.LT.MDF(3)).AND.(MDF(EVPTR).EQ.PTIME))MORE=.TRUE.
      IF((.NOT.MORE).OR.(MDF(EVPTR+1).NE.2).OR.
     +    (MDF(EVPTR+2).NE.GRNUNT).OR.(EVENT.NE.1))GO TO 50
      MUTUAL=IAND(RDUNTS,MDF(EVPTR+3))

 50   IF(EVENT.NE.1)GO TO 200
      MDF(GRNPTR+16)=IOR(MDF(GRNPTR+16),RDUNTS)
      UTCODE=1
      DO 150  I=1,NUMRED
           IF(IAND(UTCODE,RDUNTS).EQ.0)GO TO 149
           MDF(REDPTR(I)+16)=IOR(MDF(REDPTR(I)+16),GUB2)
           X1=MDF(GRNPTR+4)
           Y1=MDF(GRNPTR+5)
           X2=MDF(REDPTR(I)+4)
           Y2=MDF(REDPTR(I)+5)
           CALL GBEG(IENT,X1,Y1)
           CALL GPUT(5,1760,0,0)
           CALL GPUT(6,53,X2,Y2)
           IF(IAND(UTCODE,MUTUAL).NE.0)CALL COLOR(1)
           IENT=IENT+1
           NEW=.TRUE.
149        UTCODE=2*UTCODE
150   CONTINUE
      GO TO 900

200   IF(EVENT.NE.2)GO TO 400
      MDF(GRNPTR+16)=IOR(MDF(GRNPTR+16),RDUNTS)
      UTCODE=1
      DO 250  I=1,NUMRED
           IF(IAND(UTCODE,RDUNTS).EQ.0)GO TO 249
           MDF(REDPTR(I)+10)=IOR(MDF(REDPTR(I)+10),GUB2)
           IF(IAND(UTCODE,MUTUAL).NE.0)GO TO 249
           X1=MDF(GRNPTR+4)
           Y1=MDF(GRNPTR+5)
           X2=MDF(REDPTR(I)+4)
           Y2=MDF(REDPTR(I)+5)
           CALL GBEG(IENT,X1,Y1)
           CALL GPUT(5,1760,0,0)
           CALL GPUT(6,53,X2,Y2)
           CALL COLOR(0)
           IENT=IENT+1
           NEW=.TRUE.
           MUTUAL=0
249        UTCODE=UTCODE*2
```

```
250   CONTINUE
      GO TO 900

400   IF(EVENT.NE.-1)GO TO 600
      MOF(GRNPTR+10)=IEXOR(MOF(GRNPTR+10),RDUNTS)
      UTCODE=1
      DO 450 I=1,NUMRED
         IF(IAND(UTCODE,RDUNTS).EQ.0)GO TO 449
         MOF(REDPTR(I)+16)=IEXOR(MOF(REDPTR(I)+16),GUB2)
449      UTCODE=UTCODE*2
450   CONTINUE
      GO TO 900

600   IF(EVENT.NE.-2)GO TO 900
      MOF(GRNPTR+16)=IEXOR(MOF(GRNPTR+16),RDUNTS)
      UTCODE=1
      DO 650  I=1,NUMRED
         IF(IAND(UTCODE,RDUNTS).EQ.0)GO TO 649
         MOF(REDPTR(I)+10)=IEXOR(MOF(REDPTR(I)+10),GUB32)
649      UTCODE=UTCODE*2
650   CONTINUE

900   IF(MORE)GO TO 10


      MOF(MOF(4))=EVPTR
      CALL BLKOUTS
      IF(ITSW(3).NE.1)IDELY=100+(INDEX)*100
      IF(NEW)CALL DELAY(IDELY)
      CALL GREST(IDPL,I1,I2,I3)
      RETURN
      END
```

```
      SUBROUTINE BLKUTS
      COMMON/MDFILE/MDF(3000),MDFMAX
      IMPLICIT INTEGER(A-Z)
C---     THIS ROUTINE BLINKS THE DETECTED UNITS AND HALTS THE
C---     BLINKING OF THE REMAINING UNITS
      NUMUTS=MDF(MDF(2))+MDF(MDF(2)+1)
      UNTPTR=MDF(2)+4
      IF(NUMUTS.EQ.0)RETURN
      DO 100 K=1,NUMUTS
         CALL GENT(MDF(UNTPTR+1))
         CALL GPUT(3,130,3,0)
         IF(MDF(UNTPTR+16).NE.0)CALL GPUT(3,130,3,1)
      UNTPTR=UNTPTR+20
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TMBASE
      COMMON/PROBLM/PTIME
      COMMON/RPLAY/INDEX,INTRVL(4),ENDTIM
      IMPLICIT INTEGER(A-Z)

C  ****  CLOCK WHICH CAUSES PROBLEM TIME TO ADVANCE

      CALL DELAY(INTRVL(INDEX))
      PTIME=PTIME+1

      RETURN
      END
```

```
      SUBROUTINE DELAY(INTRVL)
      IMPLICIT INTEGER(A-Z)
C  ****  DELAYS REAL TIME RETURN TO REPLAY
      CALL TIME(MIN,MSEC)
      XTMSEC=MSEC+INTRVL
      MIN1=XTMSEC/12000
      MSEC1=XTMSEC-12000*MIN1
      MIN1=MIN1+MIN
1     CALL TIME(MIN,MSEC)
      CALL GSTT(0,0)
      IF(MSEC.LT.MSEC1.OR.MIN.LT.MIN1)GOTO1
      RETURN
      END
```

```
SUBROUTINE RESULT
RETURN
END
```

```
      SUBROUTINE MVMENT
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/UINFO/RCDSIZ
      COMMON/PROBLM/TIME
      IMPLICIT INTEGER(A-Z)
C  ****  FINDS UNITS POSITION BASED ON PROBLEM TIME, ENTER THEM INTO FILE
      UNTPTR=MDF(2)
```

```
      UPTR=UNTPTR+4
      UCOUNT=MDF(UNTPTR)+MDF(UNTPTR+1)

      DO 1 J=1,UCOUNT
      ACTPTR=MDF(UPTR+RCDSIZ-1)
      PRESNT=MDF(ACTPTR+1)+ACTPTR
      NEXT=MDF(ACTPTR+2)+ACTPTR
      CALL CALPOS(PRESNT,NEXT,X,Y,MOVCOD)
      MDF(UPTR+4)=X
      MDF(UPTR+5)=Y
      MDF(UPTR+6)=MDF(PRESNT+4)
      IF (MOVCOD.GE.0) MDF(UPTR+7)=MOVCOD
      IF(MDF(NEXT+3).GT.TIME)GOTO15
      MDF(ACTPTR)=MDF(ACTPTR+1)
      MDF(ACTPTR+1)=MDF(ACTPTR+2)
      MDF(ACTPTR+2)=MDF(ACTPTR+2)+ACTLEN(MDF(NEXT))
15    UPTR=UPTR+RCDSIZ
1     CONTINUE

      RETURN
      END
```

```
      SUBROUTINE CALPOS(PRESNT,NEXT,X,Y,MOVCOD)
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/PROBLM/TIME
      IMPLICIT INTEGER(A-Q,S-Z)

      X0=MDF(PRESNT+1)
      Y0=MDF(PRESNT+2)
      T0=MDF(PRESNT+3)
      XF=MDF(NEXT+1)
      YF=MDF(NEXT+2)
      TF=MDF(NEXT+3)
      DT=TIME-T0
      IF(T0.NE.TF)GOTO5
      X= XF
      Y=YF
      MOVCOD=-1
      GOTO 1
5     CALL VEL(X0,Y0,XF,YF,T0,TF,RVX,RVY)
      CALL IPOSIT(X0,Y0,RVX,RVY,DT,X,Y)
      MOVCOD=1
      IF (X0.EQ.XF .AND. Y0.EQ.YF) MOVCOD=0

1     RETURN
      END
```

```
      SUBROUTINE DSPLAY
      CALL POSUNT
      RETURN
      END
```

```
      SUBROUTINE CLKUPD
      COMMON/PROBLM/PTIME
      COMMON/RPLAY/INDEX,INTRVL(4),ENDTIM
      IMPLICIT INTEGER (A-Z)
      DIMENSION CODE(10)
      DATA CODE/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/

C  ****  UPDATES THE TIME NEEDLE AND NUMERIC PROBLEM TIME ON SCREEN

      CALL GENT(4)
      X=FLOAT(PTIME)*1024./FLOAT(ENDTIM)
      DX=X-500
      CALL GPUT(6,104,DX,0)

      PLACE=1000
C  ****  RETURNS HOUR:MINUTE EQUIVALENT OF MINUTE COUNTER "PTIME"
      RMDER=HRMNS(PTIME)

      DO 1 J=1,4
          NUMRAL=RMDER/PLACE*PLACE
          RMDER=RMDER-NUMRAL
          NUMRAL=NUMRAL/PLACE+1
          CALL GPUT(13+J,90,CODE(NUMRAL),0)
          PLACE=PLACE/10
1     CONTINUE

      RETURN
      END
```

```
      SUBROUTINE VEL(XO,YO,XF,YF,TO,TF,RVX,RVY)
      IMPLICIT INTEGER(A-Q,S-Z)
C
C     IXO,IYO ARE THE POSITIONS OF THE UNIT AT TIME TO.
C     IXF,IYF POSITIONS AT TIME TF
C     FROM THIS INFO CALCULATE THE COMPONENT VELOCITIES VX AND VY
C
C     DT REPRESENTS THE CHANGE IN TIME CORRESPONDING TO THE CHANGE
C     IN POSITION
C
      DT=TF-TO
      RVX=FLOAT(XF-XO)/FLOAT(DT)
      RVY=FLOAT(YF-YO)/FLOAT(DT)

      RETURN
      END
```

```
      SUBROUTINE IPOSIT(X0,Y0,RVX,RVY,DT,XP,YP)
      IMPLICIT INTEGER(A-Q,S-Z)
C
C     GIVEN INITIAL POSITIONS(IX0,IY0) COMPONENT VELOCITIES
C     OF THE UNIT(VX,VY) AND THE CHANGE IN TIME,DT, FROM THE
C     INITIAL POSITION, CALCULATE THE NEW POSITION IXP,IYP.
C
      XP=FLOAT(X0)+RVX*FLOAT(DT)+.5
      YP=FLOAT(Y0)+RVY*FLOAT(DT)+.5

      RETURN
      END
```

```
      SUBROUTINE POSUNT
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/UINFO/RCDSIZ
      IMPLICIT INTEGER (A-Z)

C  ****  POSITIONS UNITS ACCORDING TO UNIT INFORMATION RECORD

      UPTR=MDF(2)
      UTOTAL=MDF(UPTR)+MDF(UPTR+1)
      FLPTR=UPTR+4

      DO 1 I=1,UTOTAL
             ENT=MDF(FLPTR+1)
             X=MDF(FLPTR+4)
             Y=MDF(FLPTR+5)
             CALL GENT(ENT)
             CALL GPUT(1,150,X,0)
             CALL GPUT(2,110,Y,0)
           FLPTR=FLPTR+RCDSIZ
1     CONTINUE

      RETURN
      END
```

```
      SUBROUTINE TTLEGS(X1,Y1,X2,Y2,ITYPE,JLOOK,IARY,LEN)
      COMMON/MDFILE/MDF(3000),MDFMAX
C*********
C*********
C********* A PATH LEG IS PASSED IN X1,Y1,X2,Y2 AND DATA IS RETURNED IN
C********* IARY ABOUT ALL TERRAIN CROSSINGS.  TERRAIN INFO IS BIT CODED.
C*********
C********* ITYPE - MUST BE SET BY CALLING ROUTINE TO THE TERRAIN
C*********         TYPE OF X1,Y1.  IF SO SET, IARY(3) RETURNS THIS
C*********         TERRAIN TYPE MASKED BY JLOOK.
```

```
C********** JLOOK - BIT MASK USED TO INDICATE WHICH TERRAIN TYPES ARE
C**********          TO BE EXAMINED OR IGNORED WHEN LOOKING FOR CROSSINGS.
C**********          SET BITS 1 (LSB) TO 6. 1<->ROAD, 6<->FOREST.
C********** IARY - AN ARRAY IN WHICH CROSSINGS CAN BE PASSED BACK TO
C**********          THE CALLING PROGRAM.
C********** LEN - LENGTH OF IARY (NUMBER OF WORDS).
C**********
C********** IARY(1)=NUMBER OF INTERSECTIONS FOUND. OR:
C**********          IF LEN IS TOO SMALL FOR ALL OF THE POINTS TO BE FOUND
C**********         =NEGATIVE OF NUMBER OF POINTS FOUND. OR:
C**********          IF LEN IS TOO SMALL FOR ANY POINTS TO BE FOUND
C**********         =-1000
C**********
C********** IARY(3)=TERRAIN TYPE OF FIRST POINT, IN BIT CODED FORM,
C**********          MASKED BY JLOOK.
C**********
C********** IARY(5)=X OF FIRST POINT OF INTERSECTION
C********** IARY(6)=Y OF FIRST POINT OF INTERSECTION
C********** IARY(7)=ADDRESS OF TERRAIN IN THE MDF
C********** IARY(8)=TERRAIN TYPE OF FIRST POINT OF INTERSECTION
C**********          IN BIT CODED FORM.
C********** A CELL IS 5 WORDS LONG, STARTING AT IARY(4)
C********** ET CETERA
C**********
C********** IF IARY IS TOO SHORT FOR ANY POINTS TO BE RECORDED, THEN
C***********  RETURNS WITH IARY(1)=-1000
C**********
      INTEGER X1,Y1,X2,Y2,SFT
      DIMENSION INTARY(5)
      DIMENSION ITRCOD(8),IARY(1)
      DATA ITRCOD/1,2,4,8,16,32,64,128/
      IARY(2)=IAND(ITYPE,JLOOK)

      SFT=0
      LOC=5
      IF(MDF(MDF(1)).EQ.0)GOTO1000
C********** LENGTH POINTS TO THE LOCATION OF THE LAST PLACE IN IARY
C********** WHERE A SET (X,Y,TERRAIN TYPE) CAN BE PUT.
      LENGTH=LEN-4
      IF (LOC.LE.LENGTH) GO TO 4
      IARY(1)=-1000
      RETURN
4     CONTINUE

C********** SEARCH THROUGH MDF FOR CROSSINGS.
      INXT=MDF(1)
   10 IPTR=INXT
      IF(IPTR.EQ.0)GO TO 1000
      INXT=MDF(IPTR)
      IPTR1=IPTR
      NTYPE=MDF(IPTR+1)

      IF(IAND(ITRCOD(NTYPE),JLOOK).EQ.0)GO TO 10
      IF((X1.LT.MDF(IPTR+3).AND.X2.LT.MDF(IPTR+3)).OR.
     +     (Y1.LT.MDF(IPTR+4).AND.Y2.LT.MDF(IPTR+4)).OR.
     +     (X1.GT.MDF(IPTR+5).AND.X2.GT.MDF(IPTR+5)).OR.
```

```
     +     (Y1.GT.MDF(IPTR+6).AND.Y2.GT.MDF(IPTR+6))) GO TO 10
      IPTR=IPTR+5
   20 IPTR=IPTR+2
      IF(IPTR.GE.INXT-2)GO TO 10
      CALL WHEREX(X1,Y1,X2,Y2,MDF(IPTR),MDF(IPTR+1),MDF(IPTR+2),
     +     MDF(IPTR+3),INTARY)
      IF(INTARY(1).EQ.0)GO TO 20
         IF((MDF(IPTR).EQ.INTARY(2).AND.MDF(IPTR+1).EQ.INTARY(3)).OR.
     +        (MDF(IPTR+2).EQ.INTARY(2).AND.MDF(IPTR+3).EQ.INTARY(3)))
     +        IPTR=IPTR+2
         IARY(LOC)=INTARY(2)
         IARY(LOC+1)=INTARY(3)
         IARY(LOC+2)=IPTR1
         IARY(LOC+3)=ITRCOD(NTYPE)
         LOC=LOC+5
   55    IF(LOC.LE.LENGTH)GO TO 20

60    CONTINUE
C********** IARY FILLED BEFORE ALL INTERSECTIONS FOUND.
      IARY(1)=-LOC/5+1
      GO TO 1005
C********** ALL INTERSECTIONS FOUND
1000  IARY(1)=LOC/5-1
C********** SORT POINTS IN ORDER FROM X1 TO X2
1005  CONTINUE
C********** CHECK FOR ZERO POINTS
      IF(LOC.EQ.5)GOTO3000
C********** CHECK FOR ONE POINT
      IF(LOC.EQ.10)GOTO2000
C********** OK TO SORT
      LOCM5=LOC-5
      LOCM10=LOC-10
C********** SORT IN ASCENDEING ORDER?
      IF (X1.GT.X2) GO TO 1050
       IF(X1.LT.X2)GO TO 1007
       SFT=1
       IF(Y1.GT.Y2)GO TO 1050
C********** YES,ASCENDING
 1007 DO 1010 I=5,LOCM10,5
      JJ=I+5
      DO 1010 J=JJ,LOCM5,5
1010  IF(IARY(I+SFT).GT.IARY(J+SFT))CALL SWPSUM(IARY(I),IARY(J),5)
      GO TO 2000
C********** NO,DESCENDING
1050  CONTINUE
      DO 1070 I=5,LOCM10,5
      JJ=I+5
      DO 1070 J=JJ,LOCM5,5
 1070 IF(IARY(I+SFT).LT.IARY(J+SFT))CALL SWPSUM(IARY(I),IARY(J),5)
C********** COMPUTE BIT CODES FOR TYPES
2000  CONTINUE
       LOC=8
2010  IF (LOC.GT.LEN) GO TO 3000
      IARY(LOC)=IEXOR(IARY(LOC-5),IARY(LOC))
      LOC=LOC+5
      GO TO 2010
```

```
3000  RETURN
      END
```

```
      FUNCTION IWRAMI(X,Y,JLOCK)
      COMMON/MDFILE/MDF(3000),MDFMAX
      INTEGER X,Y
      DIMENSION ITRCOD(8)
      DATA ITRCOD/1,2,4,8,16,32,64,128/
      ILOCK=IAND(JLOCK,191)
C********* RETURNS A BIT CODED WORD TO INDICATE WHAT TYPE OF TERRAIN
C********* POINT X,Y IS IN.

C********* START IN CLEAR TERRAIN
      IWRAMI=0
C********* SEARCH ALL OF TERRAIN DATA FILE
      IEND=MDF(2)-1
      IPTR=MDF(1)
      IF(MDF(IPTR).EQ.0)GOTO2000
10    INXT=MDF(IPTR)
      ITYPE=MDF(IPTR+1)
C********* SOME TERRAINS ARE SKIPPED
      IF (IAND(ILOOK,ITRCOD(ITYPE)).EQ.0) GO TO 1000
C********* IS IT INSIDE THE MIN/MAX BOX?
      IF (.NOT.(
     @    X.GE.MDF(IPTR+3).AND.X.LE.MDF(IPTR+5)
     @      .AND.
     @      Y.GE.MDF(IPTR+4).AND.Y.LE.MDF(IPTR+6)
     @      )) GO TO 1000
C********* IT IS INSIDE BOX, SEE IF IT IS INSIDE BOUNDARY,
      IPTR=IPTR+7
      IF (INSIDE(X,Y,MDF(IPTR),INXT-IPTR).EQ.0) GO TO 1000
C********* IT IS INSIDE
      IWRAMI=IEXOR(IWRAMI,ITRCOD(ITYPE))
1000  IF (INXT.EQ.IEND) GO TO 2000
      IPTR=INXT
      GO TO 10
2000  RETURN
      END
```

```
      FUNCTION INSIDE(IX,IY,IXY,NXY)
C INSIDE RESTURNS 0 IF (IX,IY) IS NOT INSIDE THE CURVE SPECIFIED BY THE
C ARRAY IXY. IF THE POINT IS INSIDE THE CURVE, 1 IS RETURNED
      DIMENSION ITBL(4,4),IXY(2)
      DATA ITBL/0,-1,1000,1,1,0,-1,1000,1000,1,0,-1,-1,1000,1,0/
      INSIDE=1
      ICNT=0
      IDX=IXY(1)-IX
      IDY=IXY(2)-IY
      JGO=1
      GO TO 5000
2000  KK=NXY
```

```
      DO 1000 J=3,KK,2
      IQDL=IQD
      IDX=IXY(J)-IX
      IDY=IXY(J+1)-IY
      JGO=2
      GO TO 5000
3000  IF (ITBL(IQDL,IQD).EQ.1000) GO TO 1100
      ICNT=ICNT+ITBL(IQDL,IQD)
      GO TO 1000
5000  CONTINUE
      IF ((IDX.GE.0).AND.(IDY.GE.0)) IQD=1
      IF ((IDX.LT.0).AND.(IDY.GE.0)) IQD=2
      IF ((IDX.LT.0).AND.(IDY.LT.0)) IQD=3
      IF ((IDX.GE.0).AND.(IDY.LT.0)) IQD=4
      GO TO (2000,3000,4000),JGO
1100  IQDP=IQD
      X=IX
      Y=IY
      IF (IXY(J-1).GT.IXY(J+1))GO TO 7100
      X1=IXY(J-2)
      Y1=IXY(J-1)
      X2=IXY(J)
      Y2=IXY(J+1)
      GO TO 7200
7100  X2=IXY(J-2)
      Y2=IXY(J-1)
      X1=IXY(J)
      Y1=IXY(J+1)
7200  SM=(Y2-Y1)/(X2-X1)
      DX=X-X1
      SY=Y-Y1
      DY=SM*DX
      IF (SY.LE.DY) GO TO 7300
      IDX=X2-X
      IDY=Y1-Y
      JGO=3
      GO TO 5000
7300  IF (SY.EQ.DY) GO TO 7400
      IDX=X1-X
      IDY=Y2-Y
      JGO=3
      GO TO 5000
7400  INSIDE=0
      RETURN
4000  ICNT=ICNT+2*ITBL(IQDL,IQD)
      IQD=IQDP
1000  CONTINUE
      IF (ICNT.EQ.0) INSIDE=0
      RETURN
1200  INSIDE=0
      RETURN
      END
```

```
*** FUNCTION IOR
*** RETURNS LOGICAL OR  OF TWO INTEGERS
        TITLE    IOR
        NAME     IOR
        EXT      $SE
IOR     ENTR
        CALL $SE
        DATA     2
K       BSS      1
L       BSS      1
        LDAE*    K
        ORAE*    L
        RETU*    IOR
        END
```

```
/DASMR
*** FUNCTION IAND
*** RETURNS LOGICAL AND OF TWO INTEGERS
        TITLE    IAND
        NAME     IAND
        EXT      $SE
IAND    ENTR
        CALL $SE
        DATA     2
K       BSS      1
L       BSS      1
        LDAE*    K
        ANAE*    L
        RETU*    IAND
        END
```

```
*** FUNCTION IEXOR
*** RETURNS THE LOGICAL EXCLUSIVE OR OF TWO INTEGERS
        TITLE   IEXOR
        NAME    IEXOR
        EXT     $SE
IEXOR   ENTR
        CALL $SE
        DATA    2
K       BSS     1
L       BSS     1
        LDAE*   K
        ERAE*   L
        RETU*   IEXOR
        END
```

```
SUBROUTINE DECMIN(IDTIME,ITIME)
```

```
C  RETURNS AN ABSOLUTE MINUTE COUNTER EQUIVALENT TO A
C     DECIMAL HOUR-MINUTE NUMBER

      IHR=IDTIME/100
      MIN=IDTIME-IHR*100
      ITIME=IHR*60+MIN
      RETURN
      END
```

```
      SUBROUTINE ROAD(TPTR)
      COMMON/MDFILE/MDF(3000),MDFMAX
      IMPLICIT INTEGER (A-Q,S-W)

C  ****  CONVERTS A SIMPLE PATH SET OF ROAD SEGMENT ENDPOINTS INTO A
C  ****  CLOSED WINDING CORRIDOR'S CONTOUR SEGMENT ENDPOINTS.

      SKRPTR=MDF(3)
      FLPTR=MDF(TPTR)

      MDF(FLPTR+1)=1
      MDF(TPTR+1)=7
      MDF(FLPTR+2)=MDF(TPTR+2)+1
      DO 1 N=3,6
      MDF(FLPTR+N)=MDF(TPTR+N)
1     CONTINUE

      FLPTR=TPTR+7

C  ****  GET COORDINATES OF SEGMENT ON SIMPLE PATH SET
2     X1=MDF(FLPTR)
      Y1=MDF(FLPTR+1)
      X2=MDF(FLPTR+2)
      Y2=MDF(FLPTR+3)

C  ****  IS SLOPE DEFINED?
      IF(Y2.EQ.Y1)GOTO5
C  ****  COMPUTE SLOPE
C  ****  RMBAR IS THE PERPENDICULAR
      RMBAR=(X1-X2)/(Y2-Y1)

C  ****  COMPUTE COORDINATES OF LINES PARALLEL TO SEGMENT AND 5 UNITS AWAY
      XP=5./SQRT(RMBAR*RMBAR+1.)+X1
      YP=RMBAR*(XP-X1)+Y1
      XN=2.*X1-XP
      YN=RMBAR*(XN-X1)+Y1

      GOTO6

C  ****  SEGMENT IS VERTICLE SO JUST ADD/SUBTRACT 5 FROM X
5     XP=X1+5.
      YP=Y1
      XN=X1-5.
      YN=Y1
```

```
C  ****  SLOPE IS REAL, STORED IN  AN INTEGER ARRAY, SO MINIMIZE INTEGER
C  ****  TRUNCATION BY MULTIPLYING THE PERPENDICULAR BY 32
C  ****  HERE, WE MAKE THE SLOPE AS LARGE AS POSSIBLE FOR INTEGER WORDS
      RMBAR=32000.

C  ****  IF PATH IS GOING IN A DOWN DIRECTION WE HAVE THE LEFT PARALLEL
C  ****  LINE PRECEED THE RIGHT IN THE CONSTRUCTION OF THE CONTOUR
5     SHIFT=0
      IF(Y1.LT.Y2)SHIFT=3

C  ****  STORE THE LINES (PARALLELS) IN THE MDF SCRATCH AREA
C  ****  IN 5 WORD CELLS, WHERE THE 3RD WORD IS SLOPE AND THE FIRST
C  ****  TWO WORDS ARE THE COORDS. OF THE PRECEEDING PARALLEL AND THE
C  ****  4TH AND 5TH ARE THE RETURN PARALLEL OF THE CONTOUR.
      MDF(SKRPTR+SHIFT)=XN
      MDF(SKRPTR+SHIFT+1)=YN
      MDF(SKRPTR+2)=32000
      IF(RMBAR.NE.0.)MDF(SKRPTR+2)=(-32./RMBAR)
      MDF(SKRPTR-SHIFT+3)=XP
      MDF(SKRPTR-SHIFT+4)=YP

      SKRPTR=SKRPTR+5
      FLPTR=FLPTR+2
      IF(FLPTR.LT.MDF(TPTR)-2)GOTO2

      FLPTR=MDF(TPTR)+7
      SKREND=SKRPTR-5
      SKRPTR=MDF(3)

      ASSIGN 8 TO LOOP

C  ****  RETRIEVE THE PARAM. INFO. OF THE PRECEEDING PARALLELS
8     RM1=FLOAT(MDF(SKRPTR+2))/32.
      RM2=FLOAT(MDF(SKRPTR+7))/32.
      X1=MDF(SKRPTR)
      Y1=MDF(SKRPTR+1)
      X2=MDF(SKRPTR+5)
      Y2=MDF(SKRPTR+6)

      SKRPTR=SKRPTR+5
      IF(SKRPTR.EQ.SKREND)ASSIGN 9 TO LOOP
      GOTO12

9     ASSIGN 10 TO LOOP

C  ****  RETRIEVE THE INFO. ON THE ENDPOINT OF THE SIMPLE PATH SET
      X1=X2
      Y1=Y2
      RM1=RM2
      RM2=-(1./RM1)
      X2=MDF(MDF(TPTR)-2)
      Y2=MDF(MDF(TPTR)-1)

      SKRPTR=SKRPTR+5
      GOTO12
```

```
10    X1=MDF(SKRPTR-2)
      Y1=MDF(SKRPTR-1)
      ASSIGN 11 TO LOOP
      GOTO12

C  ****  RETRIEVE THE PARAM. INFO. ON THE RETURN PARALLELS
11    RM1=FLOAT(MDF(SKRPTR-3))/32.
      RM2=FLOAT(MDF(SKRPTR-8))/32.
      X1=MDF(SKRPTR-2)
      Y1=MDF(SKRPTR-1)
      X2=MDF(SKRPTR-7)
      Y2=MDF(SKRPTR-6)

      SKRPTR=SKRPTR+5
      IF(SKRPTR.EQ.MDF(3)+5)ASSIGN 15 TO LOOP

C  ****  FIND THE INTERSECTION POINT OF ADJACENT PARALLES
C  ****
C  ****  IF THEIR SLOPES ARE EQUAL, SELECT AND ENDPOINT AS AN INTERSECTION
12    IF(RM1.EQ.RM2)GOTO13

      RB1=Y1-RM1*X1
      RB2=Y2-RM2*X2
      X=(RB2-RB1)/(RM1-RM2)
      Y=RM1*X+RB1
      GOTO14

13    X=X1
      Y=Y1

C  ****  STUFF THE MDF WITH THE TERRAIN CONTOUR ENDPOINTS
14    MDF(FLPTR)=X
      MDF(FLPTR+1)=Y

      FLPTR=FLPTR+2

      GOTO LOOP,(8,9,10,11,15)

15    SKRPTR=SKRPTR-5
      MDF(FLPTR)=MDF(SKRPTR+3)
      MDF(FLPTR+1)=MDF(SKRPTR+4)
      MDF(FLPTR+2)=MDF(SKRPTR)
      MDF(FLPTR+3)=MDF(SKRPTR+1)
      MDF(FLPTR+4)=MDF(MDF(TPTR)+7)
      MDF(FLPTR+5)=MDF(MDF(TPTR)+8)
      FLPTR=FLPTR+6

      WRDPTR=MDF(TPTR)
      MDF(WRDPTR)=FLPTR

      CALL GEOF(MDF(TPTR+2))
      CALL DRWSGS(MDF(WRDPTR+2),MDF(WRDPTR+7),MDF(WRDPTR)-WRDPTR-7)

      RETURN
      END
```

```
      FUNCTION IOCT(X,Y,X1,Y1,X2,Y2)
C  RETURNS BINARY CODE INDICATING WHICH OF NINE AREAS THE POINT
C     (X,Y) IS LOCATED RELATIVE TO A SPECIFIED RECTANGLE

C              X,Y    :   POINT OF INTEREST
C              X1,Y1  :   LOWER LEFT CORNER OF RECTANGLE
C              X2,Y2  :   UPPER RIGHT CORNER OF RECTANGLE

      INTEGER X,Y,X1,Y1,X2,Y2
      IOCT=0
      IF (X.LT.X1) IOCT=IOCT+1
      IF (X.GT.X2) IOCT=IOCT+2
      IF (Y.LT.Y1) IOCT=IOCT+4
      IF (Y.GT.Y2) IOCT=IOCT+8
      RETURN
      END
```

```
*** FUNCTION ACTLEN(ICODE)
***    INPUT: ACTION BIT CODE ICODE
***    OUTPUT: RETURNS NUMBER OF BITS SET PLUS ONE, I.E.,
***            THE ACTION RECORD LENGTH
       TITLE   ACTLEN
       NAME    ACTLEN
       EXT     $SE
ACTLEN ENTR
       CALL    $SE
       DATA    1
ICODE  BSS     1
       LDAE*   ICODE
       TZX
       TZB
LOOP   XAN     IBR
       LRLA    1
       IXR
       SRE     SXTN,7,040
       JMP     LOOP
       IBR
       TBA
       RETU*   ACTLEN
SXTN   DATA    16
IBR    IBR
       END
```

```
      SUBROUTINE EVTINT
C---      AT FIRST PASS SET MDF(4) TO THE BEGINNING OF THE EVENT
C---      FILE, SET MDF(MDF(4)) TO THE START OF THE FIRST EVENT
```

```
COMMON/MDFILE/MDF(3000),MDFMAX
IF(MDF(4).NE.0)RETURN
MDF(4)=MDF(3)
MDF(3)=MDF(4)+3
MDF(MDF(4))=MDF(4)+3
MDF(MDF(4)+1)=0
MDF(MDF(4)+2)=0
RETURN
END
```

```
      SUBROUTINE MDFIL(ICODE)
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/FILES/FILNUM,FCB(13)
      COMMON/ENTCNT/UNTENT,REFENT
      COMMON/PROBLM/TIME
      IMPLICIT INTEGER (A-Z)
      DIMENSION DUMMY(5)

C  READS FROM OR WRITES TO DISK STORAGE OF WORKING FILES

C             ICODE :    I/O CODE
C                         1 READ
C                         2 WRITE

      CALL V$OPEN(18,18,FCB,0)
      FCB(4)=(FILNUM-1)*60+1
      DUMMY(1)=UNTENT
      DUMMY(2)=TIME
      GO TO (10,20),ICODE

10    READ(18)DUMMY,MDF
      GO TO 30

20    WRITE(18)DUMMY,MDF
30    CONTINUE

      CALL V$CLOS(18,0)
      UNTENT=DUMMY(1)
      TIME=DUMMY(2)
      RETURN
      END
```

```
      SUBROUTINE MDFSHF(TOTAL,START)
      COMMON/UINFO/RCDSIZ
      COMMON/MDFILE/MDF(3000),MDFMAX
      IMPLICIT INTEGER(A-Z)
C  ****  SHIFTS MDF TO CREATE OR DELETE SPACE, USES 'RIPPLE' SUBROUTINE
C  ****  UPDATES ALL MDF POINTERS EFFECTED BY THE SHIFT
```

```
      UNITS=MDF(2)
      FRIEND=MDF(UNITS+2)
      ENEMY=MDF(UNITS+3)
      FUCNT=MDF(UNITS)
      EUCNT=MDF(UNITS+1)
      UCOUNT=FUCNT+EUCNT
      SKRPTR=MDF(3)
      EVTPTR=MDF(4)

C  ****  FIND THE SECTION OF THE MDF THAT THE SHIFT WILL OCCUR
C  ****  AND UPDATE THE APPROPRIATE POINTERS

      IF(START.GT.UNITS)GOTO10
      MDF(2)=MDF(2)+TOTAL

10    IF(START.GT.FRIEND)GOTO20
      IF (FUCNT.EQ.0) GO TO 20
      MDF(UNITS+2)=MDF(UNITS+2)+TOTAL

20    IF(START.GT.ENEMY)GOTO30
      IF (EUCNT.EQ.0) GO TO 30
      MDF(UNITS+3)=MDF(UNITS+3)+TOTAL

30    IF (UCOUNT.EQ.0) GO TO 38
      DO 35 I=1,UCOUNT
           PTRPTR=FRIEND+I*RCDSIZ-1
           IF(START.GT.MDF(PTRPTR))GOTO35
           MDF(PTRPTR)=MDF(PTRPTR)+TOTAL
35    CONTINUE

38    IF(START.GT.SKRPTR)GOTO40
      MDF(3)=MDF(3)+TOTAL

40    IF (START.GT.EVTPTR) GO TO 50
      MDF(EVTPTR)=MDF(EVTPTR)+TOTAL
      MDF(4)=MDF(4)+TOTAL

50    CALL RIPPLE(TOTAL,START,MDF,MDFMAX)

      RETURN
      END
```

```
      SUBROUTINE RIPPLE(ITOTAL,ISTART,ARRAY,LENGTH)
      IMPLICIT INTEGER(A-Z)
      DIMENSION ARRAY(1)

C  ****  SHIFTS PORTIONS OF ARRAY UP OR DOWN TO PACK TIGHTER OR ALLOW
C     CREATION OF EXTRA SPACE

C             ITOTAL:    TOTAL NUMBER OF WORDS TO SHIFT
C                        (ITOTAL>0     SHIFT DOWN)
C                        (ITOTAL<0     SHIFT UP  )
```

```
C   ****       ISTART:   POSITION IN ARRAY TO BEGIN SHIFT
C                        (SHIFT CONTINUES TO END OF FILE)

      IF(ITOTAL) 30,60,10

C   SHIFT DOWN

10    IPTR=LENGTH-ITOTAL
      IDIFF=IPTR-ISTART+1
      IF(IDIFF .LE. 0) GO TO 50

      DO 20 I=1,IDIFF
      ARRAY(IPTR+ITOTAL)=ARRAY(IPTR)
      IPTR=IPTR-1
20    CONTINUE

      GO TO 60

C   SHIFT UP
30    IF(ISTART .LE. IABS(ITOTAL)) GO TO 50
      IPTR=ISTART
      IDIFF=LENGTH-ISTART+1

      DO 40 I=1,IDIFF
      ARRAY(IPTR+ITOTAL)=ARRAY(IPTR)
      IPTR=IPTR+1
40    CONTINUE

      GO TO 60

C   ERROR IN PARAMETERS CAUSE SHIFT TO BE IMPOSSIBLE

50    CALL GHLT
      WRITE(1,1000)ITOTAL,ISTART,LENGTH
1000  FORMAT('BAD FILE SHIFT:  TOTAL=',I5,'START=',I5,'FILE LENGTH=',I5)
      CALL GSTT(0,0)

60    RETURN
      END
```

```
      SUBROUTINE SWPSUM(IARY,JARY,NWRDS)
C********* SWAPS A SET OF DATA WITH ANOTHER SET
      DIMENSION IARY(1),JARY(1)
      DO 10 I=1,NWRDS
      JT=IARY(I)
      IARY(I)=JARY(I)
10    JARY(I)=JT
      RETURN
      END
```

```
      SUBROUTINE MOVENT(IENT1,IENT2,IDEV,IX,IY)

C  THIS ROUTINE REPOSITIONS A GIVEN ENTITY'S REFERENCE
C     POINT LOCATED IN ELEMENTS 1 AND 2
C  THE ENTITY SHOULD BE DEFINED RELATIVE TO THIS
C     REFERENCE POINT
C  THE POINT OF REPOSITION WILL BE DEFINED BY THE LOCATION
C     OF THE INDICATED DEVICE AND IS RETURNED IN THE VARIABLES
C     IX AND IY

C             IENT1 :   THE NUMBER OF THE ENTITY BEING
C                            REFERENCED BEFORE CALLING MOVENT

C             IENT2 :   THE NUMBER OF THE ENTITY THAT
C                            WILL BE REPOSITIONED

C             IDEV  :   THE DEVICE TO BE INTERROGATED
C                                +1    TRACKBALL
C                                 0    JOYSTICK
C                                -1    LIGHTPEN

C             IX,IY :   THE SCREEN COORDINATES AT WHICH
C                            THE REPOSITIONING OCCURS


      MIN=0
      MAX=1023
      CALL GENT(IENT2)
      CALL GSTT(0,0)
      IF(IDEV) 10,20,30

C  LIGHTPEN ROUTINE NOT WORKING...

10    IX=500
      IY=500
      MARGIN=0
      GO TO 40

C  JOYSTICK ROUTINE

20    CALL GDEV(IDEV,IX,IY)
      MARGIN=500
      GO TO 40

C  TRACKBALL ROUTINE

30    CALL GDEV(IDEV,IX,IY)
      IX=IX/4
      IY=IY/4
      MARGIN=500

C  CALCULATE SCREEN COORDINATES AND REPOSITION ENTITY
```

```
4     IX=IX+MARGIN
      IY=IY+MARGIN
      IF(IX .LT. MIN) IX=MIN
      IF(IX .GT. MAX) IX=MAX
      IF(IY .LT. MIN) IY=MIN
      IF(IY .GT. MAX) IY=MAX
      CALL SPOT(1,110,IX,0)
      CALL SPOT(2,110,IY,0)
      IF(IENTI .NE. 0) CALL GENT(IENTI)
      RETURN
      END
```

```
      SUBROUTINE SETPNT(IENT,IDEV,IX,IY,IKEY)

C  DESIGNATES A POINT ON THE SCREEN BY USING THE CURSOR

C              IENT :   ENTITY LAST REFERENCED BEFORE THE CALL

C              IDEV :   DEVICE TO BE INTERROGATED
C                          +1  TRACKBALL
C                           0  JOYSTICK
C                          -1  LIGHTPEN

C              IX,IY:   COORDINATES OF ACCEPTED POSITION

C              IKEY :   FUNCTION KEY TO SIGNIFY ACCEPT (0-31)

      DIMENSION KEYROW(4)
      ICURSR=1
      IROW=4-IKEY/8

      DO 10 I=1,4
      KEYROW(I)=0
      IF(I .EQ. IROW) KEYROW(I)=2**(IKEY-8*(4-I))
10    CONTINUE

      CALL LAMPS(KEYROW(1),KEYROW(2),KEYROW(3),KEYROW(4))
      CALL GSTT(0,0)
      CALL GEON(ICURSR)

C  LOCATE DESIRED POINT

20    CALL MOVENT(IENT,ICURSR,IDEV,IX,IY)
      CALL CKINT(KEY)
      IF(KEY .NE. IKEY) GO TO 20
      CALL GEOF(ICURSR)
      RETURN
      END
```

```
      SUBROUTINE SELECT(OPTION,CHOICE)
      COMMON/BRANCH/DRTURN
      IMPLICIT INTEGER (A-C,E-Z)
      LOGICAL DRTURN
      DRTURN=.FALSE.

C ****  ROUTINE ENABLES USER TO SELECT ONE OF A LIST OF OPTIONS.
C ****  OPTIONS ARE DISPLAYED IN RED ON THE LEFT SIDE OF THE SCREEN.
C ****  USER MANIPULATES A POINTING CURSOR THAT PRECEEDS THE GREEN
C ****  CHARACTERS OF THE POINTED-AT OPTION.

C ****      OPTION          NUMBER OF MEMBERS IN LIST
C ****      CHOICE          RELATIVE POSITION IN LIST OF CHOSEN OPTION

C ****  TURN ON THE LIST

      CALL ONSEL

C ****  LIGHT THE ACCEPT AND REJECT KEYS

      CALL LAMPS(64,0,0,3)

C ****  POINTER 'INC' IS 0 AND POINTS TO THE FIRST OPTION IN THE LIST

      INC=0

C ****  POSITION THE CURSOR POINTER

      CALL GENT(1)
      CALL GPUT(1,100,-399,0)
      CALL GPUT(2,110,957,0)
      CALL GEON(1)

C ****  COLOR THE FIRST ITEM IN THE LIST GREEN

      CALL GENT(1001)
      CALL GPUT(4,140,5,1)
      CALL GPUT(4,140,6,1)

C ****  POLL THE FUNC. KEYBOARD FOR AN ACCEPT/REJECT

1     CALL CKINT(KEY)

C ****  HAS THERE BEEN A REJECT

      IF(KEY.NE.0)GOTO2

      INC=INC+1
      IF(INC.EQ.OPTION)INC=0
      Y=957-INC*50

C ****  COLOR LAST ITEM RED AGAIN

      CALL GPUT(4,140,5,0)
      CALL GPUT(4,140,6,0)
```

```
C  ****  REPOSITION CURSOR

       CALL GENT(1)
       CALL GPUT(2,110,Y,0)

C  ****  COLOR NEW ITEM GREEN

       CALL GENT(1001+INC)
       CALL GPUT(4,140,5,1)
       CALL GPUT(4,140,6,1)

       GOTO1

C  ****  HAS THERE BEEN AN ACCEPT

2      IF(KEY.NE.1)GOTO3

C  ****  BEFORE RETURNING MAKE CHOSEN ITEM RED AGAIN

       CALL GPUT(4,140,5,0)
       CALL GPUT(4,140,6,0)

       CHOICE=INC+1

C  ****  TURN OFF THE LIST

       CALL OFFSEL

C  ****  FILL THE LIST WITH BLANKS

       CALL BLKSEL

       CALL GEOF(1)

       CALL LAMPS(0,0,0,0)

       RETURN

C  ****  HAS USER HIT RETURN

3      IF(KEY.NE.30)GOTO4

       CALL GPUT(4,140,5,0)
       CALL GPUT(4,140,6,0)

       CALL OFFSEL
       CALL BLKSEL
       CALL GEOF(1)
       CALL LAMPS(0,0,0,0)

       DRTURN=.TRUE.

       RETURN

4      GOTO1
```

END

```
      SUBROUTINE SELFIL(FILIO)
      COMMON/FILES/FILNUM,FCB(13)
      COMMON/BRANCH/ORTURN
      IMPLICIT INTEGER (A-C,E-Z)
      LOGICAL ORTURN

C  ****  ALLOWS USER TO SELECT A SCENARIO FILE FOR INPUT/OUTPUT (DISK)

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('SELECT A SCENARIO FILE')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('SCENARIO FILE ONE')

      CALL GSCH(1002,6)
      WRITE(15,1002)
1002  FORMAT('SCENARIO FILE TWO')

      CALL GSCH(1003,6)
      WRITE(15,1003)
1003  FORMAT('SCENARIO FILE THREE')

      CALL GSCH(1004,6)
      WRITE(15,1004)
1004  FORMAT('SCENARIO FILE FOUR')

      CALL GSTT(0,0)

C  ****  SUBROUTINE  ALLOWS USER TO SELECT DESIRED FILE
      CALL SELECT(4,FILNUM)

      IF(ORTURN)RETURN

C  ****  SUBROUTINE DOES THE REQUIRED INPUT/OUTPUT
      CALL GHLT
      CALL MDFIO(FILIO)
      CALL GSTT(0,0)

      RETURN
      END
```

```
SUBROUTINE SELMVT(TYPE)
COMMON/CLUSTR/UNTPTR(10,2),CLTYPE,XZERO,YZERO,XREF,YREF,UNTCNT
IMPLICIT INTEGER(A-Z)
```

```
C   ****  ROUTINE ALLOWS USER TO SELECT A TYPE OF MOVEMENT

      OPTNUM=5

      CALL GHLT

      CALL GSCH(1000,35)
      WRITE(15,1035)
1035  FORMAT('UNIT MOVEMENT')

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('SELECT MOVEMENT TYPE')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('SINGLE UNIT')

      CALL GSCH(1002,6)
      WRITE(15,1002)
1002  FORMAT('DIRECTION-DEPENDENT')

      CALL GSCH(1003,6)
      WRITE(15,1003)
1003  FORMAT('FIXED-POINT-DEPENDENT')

      CALL GSCH(1004,6)
      WRITE(15,1004)
1004  FORMAT('COLUMN')

      CALL GSCH(1005,6)
      WRITE(15,1005)
1005  FORMAT('LEAPFROG')

      CALL GSCH(1007,6)
      WRITE(15,1007)
1007  FORMAT('SELECT RETURN WHEN DONE')

      IF(CLTYPE.EQ.0)GOTO10
      OPTNUM=6
      CALL GSCH(1006,6)
      WRITE(15,1006)
1006  FORMAT('PREVIOUS DEFINED CLUSTER')

10    CALL GSTT(0,0)

C   ****  ALLOWS USER TO SELECT ITEM FROM ABOVE LIST
      CALL SELECT(OPTNUM,TYPE)

      RETURN
      END
```

```
      SUBROUTINE SELNOD(ENT1,ENT2,EL,CHOICE,X,Y)
      COMMON/BRANCH/DRTURN
      IMPLICIT INTEGER (A-C,E-Z)
      LOGICAL DRTURN
      DRTURN=.FALSE.

C ****  ALLOWS USER TO DRAW THE PATH OF A UNIT  USING 'WAITS' AND 'MOVES'

      OPTION=2

C ****  TURN ON THE LIST

      CALL PTHMSG
      CALL UNSEL

C ****  LIGHT THE ACCEPT AND REJECT KEYS

      CALL LAMPS(64,0,0,3)

C ****  POINTER INC IS 0 AND POINTS TO FIRST OPTION IN THE LIST

      INC=0

C ****  POSITION THE CURSOR POINTER

      CALL GENT(1)
      CALL GPUT(1,100,-399,0)
      CALL GPUT(2,110,957,0)
      CALL GEON(1)

C ****  COLOR THE FIRST ITEM IN THE LIST GREEN

      CALL GENT(1001)
      CALL GPUT(4,140,5,1)
      CALL GPUT(4,140,6,1)

C ****  POLL THE FUNCTION KEYBOARD FOR AN ACCEPT/REJECT

1     CALL CKINT(KEY)
      CALL MOVENT(ENT1,ENT2,1,X,Y)
      CALL GPUT(EL,53,X,Y)

C ****  HAS THERE BEEN A REJECT

      IF(KEY.NE.0)GOTO2

      INC=INC+1
      IF(INC.EQ.OPTION)INC=0
      Y=957-INC*50

C ****  COLOR LAST ITEM RED AGAIN

      CALL GENT(1001)
      IF(INC.EQ.0)CALL GENT(1002)
      CALL GPUT(4,140,5,0)
      CALL GPUT(4,140,6,0)
```

```
C   ****  REPOSTION CURSOR

      CALL GENT(1)
      CALL GPUT(2,110,Y,0)

C   ****  COLOR NEW ITEM GREEN

      CALL GENT(1001+INC)
      CALL GPUT(4,140,5,1)
      CALL GPUT(4,140,6,1)

      GOTO1

C   ****  HAS THERE BEEN AN ACCEPT
2     IF(KEY.NE.1)GOTO3

C   ****  BEFORE RETURNING MAKE CHOSEN ITEM RED AGAIN

      CALL GENT(1001+INC)
      CALL GPUT(4,140,5,0)
      CALL GPUT(4,140,6,0)

      CHOICE=INC+1

C   ****  TURN OFF THE LIST

      CALL OFFSEL

C   ****  FILL THE LIST WITH BLANKS

      CALL BLKSEL

      CALL GEOF(1)

      CALL LAMPS(0,0,0,0)

      RETURN

C   ****  HAS USER HIT RETURN

3     IF(KEY.NE.30)GOTO4

      CALL GENT(1001+INC)
      CALL GPUT(4,140,5,0)
      CALL GPUT(4,140,6,0)

      CALL OFFSEL
      CALL BLKSEL
      CALL GEOF(1)
      CALL LAMPS(0,0,0,0)

      DRTURN=.TRUE.

      RETURN
```

```
4       GOTO1
        END
```

```
      SUBROUTINE SELNUM(NUMBER)
      COMMON/BRANCH/DRTURN
      IMPLICIT INTEGER (A-C,E-Z)
      LOGICAL DRTURN

C  ****  ROUTINE ALLOWS USER TO CONSTRUCT A NUMERAL BETWEEN 0 AND 9999
C  ****  USING THE FUNCTION KEYBOARD IN A CALCULATOR PAD TYPE FORMAT
C  ****          (CALCULATOR TYPE PAD AT RIGHT SIDE OF KEYBOARD)

      DRTURN=.FALSE.

      CALL ONSEL
      CALL GSCH(1000,35)

10     NUMBER=0
      SUM=0
       PLACE=1000

       CALL GHLT
       WRITE(15,1000)NUMBER
       CALL GSTT(0,0)

      CALL LAMPS(71,7,7,7)

1     CALL CKINT(KEY)

       IF(PLACE.NE.0)GOTO2
       GOTO6

2     IF(.NOT.(KEY.LE.10.AND.KEY.GE.8))GOTO3
      DIGIT=11-KEY
      GOTO9

3     IF(.NOT.(KEY.LE.18.AND.KEY.GE.16))GOTO4
      DIGIT=22-KEY
      GOTO9

4     IF(.NOT.(KEY.LE.26.AND.KEY.GE.24))GOTO5
      DIGIT=33-KEY
      GOTO9

5      IF(KEY.NE.2)GOTO6
      DIGIT=0
      GOTO9

6      IF(KEY.NE.1)GOTO7
      CALL OFFSEL
```

```
        CALL BLKSEL
         CALL LAMPS(0,0,0,0)
         RETURN

7       IF(KEY.NE.0)GOTO8
         GOTO10

8       IF(KEY.NE.30)GOTO1
        UPTURN=.TRUE.
        CALL OFFSEL
        CALL BLKSEL
        CALL LAMPS(0,0,0,0)
        RETURN

9       SUM=SUM+DIGIT*PLACE

        NUMBER=SUM/PLACE
        CALL GHLT
        WRITE(15,1000)NUMBER
        CALL GSTT(0,0)

        PLACE=PLACE/10
         IF(PLACE.EQ.0)CALL LAMPS(0,0,0,3)
        GOTO1

1000    FORMAT(I5)

        END
```

```
      SUBROUTINE SELPOS(POSTUR)
      IMPLICIT INTEGER (A-Z)

C  ****  ALLOWS USER TO SELECT A UNIT COMBAT POSTURE

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('SELECT A COMBAT POSTURE')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('DEFENSIVE')

      CALL GSCH(1002,6)
      WRITE(15,1002)
1002  FORMAT('OFFENSIVE')

      CALL GSTT(0,0)

C  ****  ALLOWS USER TO SELECT ITEM IN ABOVE LIST
      CALL SELECT(2,POSTUR)
```

RETURN
END

```
      SUBROUTINE SELPSP(PRCNTG)
      COMMON/BRANCH/DRTURN
      IMPLICIT INTEGER (A-C,E-Z)
      LOGICAL DRTURN

C  ****  ALLOWS USER TO SELECT A PATH SPEED PERCENTAGE

      CODE=0

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('SELECT A PATH SPEED')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('MAXIMUM')

      CALL GSCH(1002,6)
      WRITE(15,1002)
1002  FORMAT('NOMINAL')

      CALL GSCH(1003,6)
      WRITE(15,1003)
1003  FORMAT('PERCENTAGE MAXIMUM')

      CALL GSCH(1004,6)
      WRITE(15,1004)
1004  FORMAT('NODE MODE')

      CALL GSTT(0,0)

C  ****  ALLOWS USER TO SELECT ITEM IN ABOVE LIST
2     CALL SELECT(4,CODE)
      IF(DRTURN)RETURN

      IF(CODE.NE.3)GOTO5
3     CALL SELNUM(PRCNTG)
      IF(DRTURN)GOTO2
      IF(PRCNTG.GT.100)GOTO3
      RETURN

5     IF(CODE.NE.2)GOTO6
      PRCNTG=80
      RETURN

6     IF (CODE.NE.1)GOTO7
      PRCNTG=100
      RETURN
```

```
7       PRCNTG=-1
        RETURN

        END
```

```
      SUBROUTINE SELREF(PRCDRE)
      IMPLICIT INTEGER (A-Z)

C  ****  ALLOWS USER TO SELECT TO DRAW OR DELETE REFERENCE LINES

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('SELECT DESIRED PROCEDURE')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('CONSTRUCT REFERENCE LINES')

      CALL GSCH (1002,6)
      WRITE(15,1002)
1002  FORMAT('ERASE REFERENCE LINES')

      CALL GSTT(0,0)

C  ****  SUBRCUTNE ALLOWS USER TO SELECT ITEM IN ABOVE LIST
      CALL SELECT(2,PRCDRE)

      RETURN
      END
```

```
      SUBROUTINE SELSID(SIDE)
      IMPLICIT INTEGER(A-Z)

C  ****  ALLOWS USER TO SELECT UNIT FORCE, FRIENDLY OR ENEMY

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('SELECT UNIT FORCE')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('FRIENDLY')

      CALL GSCH(1002,6)
      WRITE(15,1002)
```

```
1002  FORMAT('ENEMY')

      CALL GSTT(0,0)

C  ****  SUBROUTINE ALLOWS USER TO SELECT ITEM IN ABOVE LIST
      CALL SELECT(2,SIDE)

      RETURN
      END
```

```
      SUBROUTINE SELSPD(PRCNTG)
      COMMON/BRANCH/DRTURN
      IMPLICIT INTEGER (A-C,E-Z)
      LOGICAL DRTURN

C  ****  ALLOWS USER TO SELECT A UNIT SPEED PERCENTAGE

      CODE=0

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('SELECT A UNIT SPEED')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('MAXIMUM')

      CALL GSCH(1002,6)
      WRITE(15,1002)
1002  FORMAT('NOMINAL')

      CALL GSCH(1003,6)
      WRITE(15,1003)
1003  FORMAT('PERCENTAGE MAXIMUM')

      CALL GSTT(0,0)

C  ****  ALLOWS USER TO SELECT ITEM IN ABOVE LIST
2     CALL SELECT(3,CODE)
      IF(DRTURN)RETURN

      IF(CODE.NE.3)GOTO5
3     CALL SELNUM(PRCNTG)
      IF(DRTURN)GOTO2
      IF(PRCNTG.GT.100)GOTO3
      RETURN

5     IF(CODE.NE.2)GOTO6
      PRCNTG=80
      RETURN
```

```
6       PRCNTG=100
        RETURN
        END
```

```
      SUBROUTINE SELTER(TTYPE)
      IMPLICIT INTEGER (A-Z)

C  ****  ALLOWS USER TO SELECT A TERRAIN FOR DEFINITION

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('SELECT A TERRAIN TYPE')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('ROAD')

      CALL GSCH(1002,6)
      WRITE(15,1002)
1002  FORMAT('RIVER')

      CALL GSCH(1003,6)
      WRITE(15,1003)
1003  FORMAT('LAKE')

      CALL GSCH(1004,6)
      WRITE(15,1004)
1004  FORMAT('CITY')

      CALL GSCH(1005,6)
      WRITE(15,1005)
1005  FORMAT('HILL')

      CALL GSCH(1006,6)
      WRITE(15,1006)
1006  FORMAT('FOREST')

      CALL GSCH(1007,6)
      WRITE(15,1007)
1007  FORMAT('RETURN WHEN COMPLETED')

      CALL GSTT(0,0)

C  ****  SUBROUTINE ALLOWS USER TO SELECT DESIRED TYPE
      CALL SELECT(6,TTYPE)

      RETURN
      END
```

```
      SUBROUTINE SELTIM(TIME)
      COMMON/PROBLM/PTIME
      COMMON/BRANCH/DRTURN
      COMMON/RPLAY/INDEX,INTRVL(4),ENDTIM
      COMMON/MDFILE/MDF(3000),MDFMAX
      IMPLICIT INTEGER (A-Z)
      LOGICAL DRTURN

C  ****  ALLOWS THE USER TO SPECIFY A TIME WHICH IS LESS THAN OR
C  ****  EQUAL TO THE PROBLEM TIME

1     CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,901)
901   FORMAT('ENTER NEW VALUE')

      CALL GSCH(1001,6)
      WRITE(15,902)
902   FORMAT('FOR PROBLEM TIME')

      CALL GSTT(0,0)

      CALL SELNUM(HMTIM)
      IF (DRTURN) RETURN
      CALL DECMIN(HMTIM,TIME)
      IF(TIME.GT.ENDTIM)GO TO 1

      RETURN
      END
```

```
      SUBROUTINE SELUNT(UNTPTR)
      COMMON/MDFILE/MDF(3000),MDFMAX
      COMMON/UINFO/RCDSIZ
      COMMON/BRANCH/DRTURN
      IMPLICIT INTEGER(A-C,E-Z)
      LOGICAL DRTURN
C  ****  ROUTINE ALLOWS THE USER TO SELECT A UNIT WITH THE TRACKBALL
      UNTENT=0
      TBCRCL=5
      TBALL=1
      ACCEPT=1
      RTURN=30
      UPTR=MDF(2)
      UCOUNT=MDF(UPTR)+MDF(UPTR+1)
      UCNTM1=UCOUNT-1
      FELPTR=UPTR+4
      DRTURN=.FALSE.
```

```
      CALL GEON(TBCRCL)
      CALL SELUMG
      CALL ONSEL

1     CALL MOVENT(UNTENT,TBCRCL,TBALL,X,Y)
      CALL CKINT(KEY)

      UNTPTR=0
      DLTAX1=25
      DLTAY1=25

      DO 2 J=0,UCNTM1
            UPTR=FFLPTR+J*RCDSIZ
            UNITX=MDF(UPTR+4)
            UNITY=MDF(UPTR+5)
            DELTAX=IABS(X-UNITX)
            DELTAY=IABS(Y-UNITY)
            IF(DELTAX.GT.25.OR.DELTAY.GT.25)GOTO2
            IF(DELTAX.GT.DLTAX1.OR.DELTAY.GT.DLTAY1)GOTO2
            DLTAX1=DELTAX
            DLTAY1=DELTAY
            UNTPTR=UPTR
2     CONTINUE

      IF(UNTPTR.NE.0)GOTO3
      CALL GPUT(3,130,3,0)
      UNTENT=0
      CALL LAMPS(64,0,0,0)
      GOTO5

3     ENTNUM=MDF(UNTPTR+1)
      IF(UNTENT.EQ.ENTNUM)GOTO4

      CALL GPUT(3,130,3,0)
      UNTENT=ENTNUM
      CALL GENT(UNTENT)
      CALL GPUT(3,130,3,1)

4     CALL LAMPS(64,0,0,2)
      IF(KEY.NE.ACCEPT)GOTO5
      CALL GPUT(3,130,3,0)
      CALL GEOF(TBCRCL)
      CALL OFFSEL
      CALL BLKSEL
      RETURN

5     IF(KEY.NE.RTURN)GOTO1
      DRTURN=.TRUE.
      CALL GPUT(3,130,3,0)
      CALL GEOF(TBCRCL)
      CALL OFFSEL
      CALL BLKSEL
      RETURN

      END
```

```
      SUBROUTINE SELUSP(PRCDRE)
      COMMON/PROBLM/TIME
      IMPLICIT INTEGER (A-Z)

C  ****  ALLOWS USER TO SELECT A UNIT STATE MANIPULATION PROCEDURE

      OPTNS=4

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('SELECT A PROCEDURE')

      CALL GSCH(1000,35)
      WRITE(15,1111)
1111  FORMAT('UNIT STATE')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('ADD')

      CALL GSCH(1002,6)
      WRITE(15,1002)
1002  FORMAT('DELETE')

      CALL GSCH(1003,6)
      WRITE(15,1003)
1003  FORMAT('RESUPPLY')

      CALL GSCH(1004,6)
      WRITE(15,1004)
1004  FORMAT('COMBAT POSTURE')

      IF(TIME.NE.0)GOTO2
      CALL GSCH(1005,6)
      WRITE(15,1005)
1005  FORMAT('RELOCATE')

      OPTNS=5

2       CALL GSTT(0,0)
C  ****  SUBROUTINE ALLOWS USER TO SELECT ITEM IN ABOVE LIST
        CALL SELECT(OPTNS,PRCDRE)

      RETURN
      END
```

```
SUBROUTINE SELUTS(SIZE)
IMPLICIT INTEGER(A-Z)
```

```
C   ****  ALLOWS USER TO SELECT A UNIT SIZE

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('SELECT UNIT SIZE')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('BRIGADE')

      CALL GSCH(1002,6)
      WRITE(15,1002)
1002  FORMAT('BATTALION')

      CALL GSCH(1003,6)
      WRITE(15,1003)
1003  FORMAT('COMPANY')

      CALL GSTT(0,0)

      CALL SELECT(3,SIZE)

      RETURN
      END
```

```
      SUBROUTINE SELUTT(TYPE)
      IMPLICIT INTEGER (A-Z)

C  ****  ALLOWS USER TO SELECT A UNIT TYPE

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('SELECT UNIT TYPE')

      CALL GSCH(1001,6)
      WRITE(15,1002)
1002  FORMAT('ARTILLERY')

      CALL GSCH(1002,6)
      WRITE(15,1001)
1001  FORMAT('ARMOR')

      CALL GSCH(1003,6)
      WRITE(15,1003)
1003  FORMAT('INFANTRY')

      CALL GSCH(1004,6)
      WRITE(15,1004)
```

```
1004  FORMAT('MECH. INFANTRY')

      CALL GSCH(1006,0)
      WRITE(15,1006)
1006  FORMAT('RETURN WHEN COMPLETE')

      CALL GSTT(0,0)

C  ****  SUBROUTINE ALLOWS USER TO SELECT DESIRED TYPE OF UNIT
2     CALL SELECT(4,TYPE)

      RETURN
      END
```

```
      SUBROUTINE ADUMSG
      COMMON/PROBLM/TIME
      IMPLICIT INTEGER (A-Z)

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('PLACE UNIT WITH TRACKBALL')

      IF(TIME.EQ.0)GOTO1
      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('ON SCENARIO BOUNDARY')

1     CALL GSTT(0,0)
      RETURN
      END
```

```
      SUBROUTINE BPTHMG
      IMPLICIT INTEGER(A-Z)

      CALL GHLT

      CALL GSCH(1000,0)
      WRITE(15,1000)
 1000 FORMAT('UNIT IMMOBILE AT POSITION')

      CALL GSCH(1001,0)
      WRITE(15,1001)
 1001 FORMAT('INDICATED ON PATH')

      CALL GSTT(0,0)

      RETURN
      END
```

```
      SUBROUTINE CLCMSG
      IMPLICIT INTEGER (A-Z)

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('LOCATE A CLUSTER CENTER')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('WITH TRACKBALL/ACCEPT KEY')

      CALL GSTT(0,0)

      RETURN
      END
```

```
      SUBROUTINE CLFMSG(CODE)
      IMPLICIT INTEGER (A-Z)

      CALL GHLT

      CALL GSCH(1004,6)
      WRITE(15,1004)
1004  FORMAT('YOU MAY REJECT UNITS')

      CALL GSCH(1005,6)
      GOTO (10,20),CODE
10    WRITE(15,1005)
1005  FORMAT('MAXIMUM UNITS SELECTED')
      GOTO30
20    WRITE(15,1006)
1006  FORMAT(' ')

30    CALL GSTT(0,0)

      RETURN
      END
```

```
SUBROUTINE CNNMSG
IMPLICIT INTEGER(A-Z)

CALL GHLT

CALL GSCH(10[illegible],5)
```

```
      WRITE(15,1000)
1000  FORMAT('INPUT NUMBER OF CONTOURS')

      CALL GSTT(0,0)

      RETURN
      END
```

```
      SUBROUTINE DFTRMS(CLASS)
      IMPLICIT INTEGER (A-Z)

      CALL GHLT

      GOTO (2,1),CLASS

1     CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('CONSTRUCT CONTOUR WITH')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('TRACKBALL AND ACCEPT KEY')

      CALL GSTT(0,0)

      RETURN

2     CALL GSCH(1000,6)
      WRITE(15,1004)
1004  FORMAT('CONSTRUCT PATH WITH')

      CALL GSCH(1001,6)
      WRITE(15,1001)

      CALL GSCH(1002,6)
      WRITE(15,1003)
1003  FORMAT('LAST NODE SELECT RETURN')

      CALL GSTT(0,0)

      RETURN
      END
```

```
SUBROUTINE DSTMSG
IMPLICIT INTEGER(A-Z)

CALL GHLT

CALL GSCH(1000,6)
WRITE(15,1000)
```

```
1000  FORMAT('LOCATE A DESTINATION')

      CALL GSCH(1001,0)
      WRITE(15,1001)
1001  FORMAT('WITH THE TRACKBALL')

      CALL GSTI([illegible],0)

      RETURN
      END
```

```
      SUBROUTINE FORCES(ISIZE,ITYPE,IEL)

C   DRAWS A PICTURE REPRESENTING A UNIT AS DESCRIBED
C      BY THE PARAMETER LIST

C             ISIZE :   THE SIZE OF THE UNIT
C                                            1  BRIGADE
C                                            2  BATTALION
C                                            3  COMPANY

C             ITYPE :   THE TYPE OF THE UNIT
C                                            1  ARTILLERY
C                                            2  ARMOR
C                                            3  INFANTRY
C                                            4  MECH. INFANTRY

      I=8

      CALL GPUT(IEL,1740,0,1)
      IEL=IEL+1
      CALL GPUT(IEL,70,2*I,1)
      IEL=IEL+1
      CALL GPUT(IEL,51,-4*I,0)
      IEL=IEL+1
      CALL GPUT(IEL,52,-2*I,0)
      IEL=IEL+1
      CALL GPUT(IEL,51,4*I,0)
      IEL=IEL+1
      CALL GPUT(IEL,52,2*I,0)
      IEL=IEL+1
      CALL GPUT(IEL,70,-2*I,-1)
      IEL=IEL+1

C   DRAW CORRECT SYMBOL

      GOTO(80,70,60),ISIZE

C     COMPANY SYMBOL

   60 CONTINUE
      CALL GPUT(IEL,72,I,0)
```

```
      IEL=IEL+1
      CALL GPUT(IEL,52,1,0)
      IEL=IEL+1
      CALL GPUT(IEL,72,-2*I,0)
      IEL=IEL+1
      GO TO 100

C     BATTALION SYMBOL

   70 CONTINUE
      CALL GPUT(IEL,70,-3,I)
      IEL=IEL+1
      CALL GPUT(IEL,52,I,0)
      IEL=IEL+1
      CALL GPUT(IEL,70,6,-I)
      IEL=IEL+1
      CALL GPUT(IEL,52,I,0)
      IEL=IEL+1
      CALL GPUT(IEL,70,-3,-2*I)
      IEL=IEL+1
      GO TO 100

C     BRIGADE SYMBOL

   80 CONTINUE
      CALL GPUT(IEL,70,-5,I+1)
      IEL=IEL+1
      CALL GPUT(IEL,1750,1,1)
      IEL=IEL+1
      CALL GPUT(IEL,90,1HX,0)
      IEL=IEL+1
      CALL GPUT(IEL,1740,0,1)
      IEL=IEL+1
      CALL GPUT(IEL,70,-9,-I-1)
      IEL=IEL+1

  100 CONTINUE
      GO TO (110,120,130,130),ITYPE

C     ARTILLERY SYMBOL

  110 CONTINUE
      CALL GPUT(IEL,70,-6,-12)
      IEL=IEL+1
      CALL GPUT(IEL,1750,1,1)
      IEL=IEL+1
      CALL GPUT(ILL,90,1H*,0)
      IEL=IEL+1
      CALL GPUT(IEL,1740,0,1)
      IEL=IEL+1
      CALL GPUT(IEL,70,-8,12)
      IEL=IEL+1
      GO TO 1000

C     ARMORED SYMBOL
```

```
  120 CONTINUE
      CALL GPUT (IEL,70,12,-9)
      IEL=IEL+1
      CALL GPUT(IEL,1754,2,1)
      IEL=IEL+1
      CALL GPUT(IEL,90,1H0,0)
      IEL=IEL+1
      CALL GPUT(IEL,1740,0,1)
      IEL=IEL+1
      CALL GPUT(IEL,70,-13,13)
      IEL=IEL+1
      GO TO 1000

C     INFANTRY SYMBOL

  130 CONTINUE
      CALL GPUT(IEL,130,1,2)
      IEL=IEL+1
      CALL GPUT(IEL,50,2*I,I)
      IEL=IEL+1
      CALL GPUT(IEL,71,-4*I,0)
      IEL=IEL+1
      CALL GPUT(IEL,53,4*I,-2*I)
      IEL=IEL+1
      CALL GPUT(IEL,71,-4*I,0)
      IEL=IEL+1
      CALL GPUT(IEL,50,2*I,I)
      IEL=IEL+1
      CALL GPUT(IEL,130,1,0)
      IEL=IEL+1
      IF (ITYPE.EQ.3) GO TO 1000

C     MECH. INFANTRY SYMBOL

      CALL GPUT(IEL,70,-5,-I-15)
      IEL=IEL+1
      CALL GPUT(IEL,1750,1,1)
      IEL=IEL+1
      CALL GPUT(IEL,90,1HM,0)
      IEL=IEL+1
      CALL GPUT(IEL,1740,0,1)
      IEL=IEL+1
      CALL GPUT(IEL,70,-9,I+15)
      IEL=IEL+1

 1000 CONTINUE
      RETURN
      END
```

```
SUBROUTINE LDUMSG
IMPLICIT INTEGEF (A-Z)

CALL GHLT
```

```
      CALL GSCH(1004,6)
      WRITE(15,1004)
 1004 FORMAT('SELECT LEAD UNIT')

      CALL GSTT(0,0)

      RETURN
      END
```

```
      SUBROUTINE MBTBL
      IMPLICIT INTEGER (A-Z)
      REAL ROUND
      DIMENSION ENTNUM(4),SPEED(7,4)

      DATA SPEED/0,1,-5,6,-14,15,-25,0,1,-5,6,-14,15,-25,
     +   0,0,-1,1,-2,3,-4,0,1,-5,6,-14,15,-25/
      DATA ENTNUM/2303,2304,2305,2306/


      DO 1 I=1,4

      CALL GBEG(ENTNUM(I),1000,1000)
      CALL GEOF(ENTNUM(I))
      CALL COLOR(2)
      CALL GPUT(5,104,30,0)

      CALL GCHA(ENTNUM(I),6,0,2,23)

      CALL GHLT
      GOTO(10,20,30,35),I

10    WRITE(15,100)
100   FORMAT('     ARTILLERY MOBILITY')
      GOTO40

20    WRITE(15,200)
200   FORMAT('        ARMOR MOBILITY')
      GOTO40

30    WRITE(15,300)
300   FORMAT('      INFANTRY MOBILITY')
      GO TO 40

35    WRITE(15,350)
350   FORMAT(' MECH. INFANTRY MOBILITY')

40    CONTINUE
      CALL GSTT(0,0)

      CALL GPUT(35,104,-568,0)
      CALL GPUT(36,114,-30,0)
```

```
      CALL GCHA(ENTNUM(I),37,0,1,40)

      CALL GHLT
      WRITE(15,500)
500   FORMAT('          SPEED RANGES (KPH)')
      CALL GSTT(0,0)

      CALL GPUT(78,104,-560,0)
      CALL GPUT(79,114,-30,0)

      CALL GCHA(ENTNUM(I),80,0,1,40)

      CALL GHLT
      WRITE(15,600)
600   FORMAT('       NO-GO  V.SLO-GO  SLO-GO      GO')
      CALL GSTT(0,0)

      EL=121

      DO 70 J=1,3
            CALL GPUT(EL,100,1000,0)
            EL=EL+1
            CALL GPUT(EL,104,44,0)
            EL=EL+1
            CALL GPUT(EL,110,940-J*50,0)
            EL=EL+1
            CALL FORCES(4-J,I,EL)
            IF (I.EQ.1) GO TO 75
70    CONTINUE

75    CONTINUE

      DO 80 J=1,3

            CALL GPUT(EL,100,1000,0)
            EL=EL+1
            CALL GPUT(EL,104,30,0)
            EL=EL+1
            CALL GPUT(EL,110,933-J*50,0)
            EL=EL+1

            CALL GCHA(ENTNUM(I),EL,0,1,40)

            CALL GHLT
            WRITE(15,800)(SPEED(K,I),K=1,7)
800         FORMAT(7X,I2,4X,2I3,3X,2I3,3X,2I3)
            CALL GSTT(0,0)

            EL=EL+41

            DO 801 K=1,7
                ROUND=0.5
                IF (SPEED(K,I).LT.0) ROUND=-0.5
                SPEED(K,I)=IFIX(FLOAT(SPEED(K,I))*.80+ROUND)
801         CONTINUE
```

```
      IF (I.EQ.1) GO TO 85
8?    CONTINUE
85    CONTINUE
1     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MESAGE(I)
      COMMON/DISPL/IDPL(4900),IERR

      CALL GHLT
      CALL GSCH(1000,35)
      IF(I.EQ.0)WRITE(15,1000)
      IF(I.EQ.1)WRITE(15,2000)
      CALL GSCH(1001,6)
      WRITE(15,1001)
      CALL GSCH(1002,6)
      WRITE(15,1002)
      CALL GSCH(1003,6)
      WRITE(15,1003)
      CALL GSCH(1004,6)
      IF(I.EQ.0)WRITE(15,1004)
      IF(I.EQ.1)WRITE(15,2004)
      CALL GSCH(1005,6)
      WRITE(15,1005)
1000  FORMAT('NOTE')
2000  FORMAT('WARNING')
1001  FORMAT('CURRENT REPLAY HAS')
1002  FORMAT('NOT ESTABLISHED')
1003  FORMAT('EVENTS OCCURRING')
2004  FORMAT('AT THIS PROBLEM')
1004  FORMAT('BEYOND THIS PROBLEM')
1005  FORMAT('TIME')
      CALL DNSEL
      CALL GSTT(0,0)
      RETURN
      END
```

```
      SUBROUTINE ONSEL
      IMPLICIT INTEGER (A-Z)
C  ****  TURNS ON CHARACTER AREAS AT THE LEFT SIDE OF THE SCREEN
      DO 1 ENTITY=1000,1010
      CALL GEON(ENTITY)
1     CONTINUE
```

FE TURN
END

```
      SUBROUTINE  OFFSEL
      IMPLICIT INTEGER (A-Z)
C  ****  TURNS OFF CHARACTER AREAS ON THE LEFT SIDE OF THE SCREEN
      DO 1 ENTITY=1000,1010
      CALL GEOF(ENTITY)
1     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BLKSEL
      IMPLICIT INTEGER (A-Z)
C  ****  FILLS CHARACTER AREAS AT LEFT SIDE OF SCREEN WITH BLANKS
      CALL GHLT
      DO 1 ENTITY=1000,1010
      CALL GSCH(ENTITY,6)
      WRITE(15,1000)
1000  FORMAT(' ')
1     CONTINUE
      CALL GSCH(1000,35)
      WRITE(15,1000)
      CALL GSTT(0,0)
      RETURN
      END
```

```
      SUBROUTINE OOBMSG(SIDE)
      IMPLICIT INTEGER(A-Z)

      CALL GSCH(1[illegible],35)

      CALL GHLT

      GOTO(1,2),SIDE

1     WRITE(15,1000)
1000  FORMAT('FRIENDLY ORDER OF BATTLE')
```

```
      GOTO 3

2     WRITE(15,2000)
2000  FORMAT('ENEMY ORDER OF BATTLE')

3     CALL GSTT(0,0)

      RETURN
      END



      SUBROUTINE PTHMSG
      IMPLICIT INTEGER (A-Z)

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('DRAW MOVEMENT PATH')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('MOVE')

      CALL GSCH(1002,6)
      WRITE(15,1002)
1002  FORMAT('WAIT')

      CALL GSCH(1003,6)
      WRITE(15,1003)
1003  FORMAT('SELECT RETURN WHEN DONE')

      CALL GSCH(1005,6)
      WRITE(15,1005)
1005  FORMAT('MAX OF 10 LEGS')

      CALL GSTT(0,0)

      RETURN
      END
```

```
      SUBROUTINE RCMSG
      IMPLICIT INTEGER(A-Z)

      CALL GHLT

      CALL GSCH(1[illegible],55)
      WRITE(15,1000)
1000  FORMAT('FUTURE POSITION CONTOURS')

      CALL GSTT(0,0)
```

```
      RETURN
      END

      SUBROUTINE PC3MSG
      IMPLICIT INTEGER(A-Z)

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('TO VIEW ONE CONTOUR AT A')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('TIME, SELECT REJECT')

      CALL GSCH(1002,6)
      WRITE(15,1002)
1002  FORMAT('SELECT RETURN WHEN DONE')

      CALL GSTT(0,0)

      RETURN
      END
```

```
      SUBROUTINE REFMSG(PRCDRE)
      IMPLICIT INTEGER (A-Z)

      CALL GHLT

      GOTO (1,2),PRCDRE

1     CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('CONSTRUCT REFERENCE LINE')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('WITH TRACKBALL/ACCEPT KEY')

      CALL GSCH(1002,6)
      WRITE(15,1002)
1002  FORMAT('SELECT RETURN WHEN LINE')

      CALL GSCH(1003,6)
      WRITE(15,1003)
1003  FORMAT('IS COMPLETE')

      CALL GSCH(1004,6)
      WRITE(15,1004)
```

```
1004  FORMAT('SELECT RETURN AGAIN AFTER')

      CALL GSCH(1005,6)
      WRITE(15,1005)
1005  FORMAT('LAST LINE IS DRAWN')

      CALL GSTT(0,0)

      RETURN

2     CALL GSCH(1000,6)
      WRITE(15,1006)
1006  FORMAT('SELECT REFERENCE LINES')

      CALL GSCH(1001,6)
      WRITE(15,1007)
1007  FORMAT('TO BE ERASED WITH THE')

      CALL GSCH(1002,6)
      WRITE(15,1008)
1008  FORMAT('LIGHTPEN AND ACCEPT KEY')

      CALL GSTT(0,0)

      RETURN

      END
```

```
      SUBROUTINE REFPTMG
      IMPLICIT INTEGER(A-Z)

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('SELECT A REFERENCE POINT')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('WITH TRACKBALL/ACCEPT KEY')

      CALL GSTT(0,0)

      RETURN
      END
```

```
SUBROUTINE RSPMSG
IMPLICIT INTEGER (A-Z)

CALL GHLT
```

```
      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('ENTER A RESUPPLY')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('PERCENTAGE OF MAXIMUM')

      CALL GSCH(1003,6)
      WRITE(15,1003)
1003  FORMAT('ACCEPT/REJECT LEVEL')

      CALL GSTT(0,0)

      RETURN
      END
```

```
      SUBROUTINE SELUMG
      IMPLICIT INTEGER(A-Z)

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000   FORMAT('SELECT UNIT WITH THE')

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('TRACKBALL AND ACCEPT KEY')

      CALL GSCH(1002,6)
      WRITE(15,1002)
1002  FORMAT('RETURN WHEN COMPLETED')

      CALL GSTT(0,0)

      RETURN
      END
```

```
      SUBROUTINE TCOLOR(ITYPE)
C  COLORS CONTOUR FEATURES
      GOTO(10,20,20,30,40,50,10,40),ITYPE
C  ROAD -- GREEN LINE
10    IBET=0
      GO TO 70
```

```
C  LAKE AND RIVER -- RED LINE

20      I=0
        J=0
        IBRT=3
        GO TO 60

C  CITY -- ORANGE LINE

30      I=1
        J=0
        IBRT=3
        GO TO 60

C  HILL -- YELLOW LINE

40      I=0
        J=1
        IBRT=1
        GO TO 60

C  FOREST -- GREEN DASH

50      CALL GPUT(3,130,1,2)
        IBRT=1
        GO TO 70

60      CALL GPUT(4,140,5,I)
        CALL GPUT(4,140,6,J)
70      CALL GPUT(3,130,2,IBRT)
        RETURN
        END
```

```
      SUBROUTINE TMS3(TIMES)
      COMMON/MOVE/MOVENT
      IMPLICIT INTEGER(A-Z)
      COMMON ARRAY(63)
      DIMENSION TIMES(1)
C  ****  DISPLAYS ARRIVAL TIMES CONTAINED IN TIMES AT NODES IN ARRAY
      XOLD=ARRAY(2)
      YOLD=ARRAY(3)
      PNTCNT=ARRAY(1)
      DO 10 I=1,PNTCNT
            X=ARRAY(I*2+2)-42
            Y=ARRAY(I*2+3)-7
            IF(X.EQ.XOLD.AND.Y.EQ.YOLD)Y=Y-15
            ENT=MOVENT+I
            CALL GEON(ENT)
            CALL GENT(ENT)
            CALL GPUT(1,100,X,0)
```

```
            CALL GPUT(2,110,Y,0)
            CALL GSCH(ENT,5)
            OUTPUT=HRMNS(TIMES(I))
            CALL GHLT
            WRITE(15,1000)OUTPUT
            CALL GSTT(0,0)
            XOLD=X
            YOLD=Y
1[illegible]    CONTINUE

      RETURN
1000  FORMAT(I4)
      END
```

```
      SUBROUTINE TRNBRT(CODE)
      COMMON/MDFILE/MDF(3000),MDFMAX
      IMPLICIT INTEGER(A-Z)

C  **** SETS TERRAIN BRIGHTNESS TO CHOSEN LEVEL  (CODE)

      FLPTR=1
1     FLPTR=MDF(FLPTR)
      IF(MDF(FLPTR).EQ.0)RETURN

      CALL GENT(MDF(FLPTR+2))
      CALL GPUT(3,130,2,CODE)

      GOTO1

      END
```

```
      SUBROUTINE TRRNID(TYPE)
      IMPLICIT INTEGER(A-Z)

      CALL GSCH(1000,35)

      CALL GHLT

      GOTO(1,2,3,4,5,6,7,8),TYPE

1     WRITE(15,10)
10    FORMAT('ROAD')
      GOTO90

2     WRITE(15,20)
20    FORMAT('RIVER')
      GOTO90

3     WRITE(15,30)
30    FORMAT('LAKE')
```

```
      GOTO90

4     WRITE(15,40)
40    FORMAT('CITY')
      GOTO90

5     WRITE(15,50)
50    FORMAT('HILL')
      GOTO90

6     WRITE(15,60)
60    FORMAT('FOREST')
      GOTO90

7     WRITE(15,70)
70    FORMAT(' ')
      GOTO90

8     WRITE(15,80)
80    FORMAT('INNER HILL')

90    CALL GSTT(0,0)
      RETURN

      END
```

```
      SUBROUTINE UPSMSG
      IMPLICIT INTEGER(A-Z)

      CALL GHLT

      CALL GSCH(1000,6)
      WRITE(15,1000)
1000  FORMAT('PLACE UNIT WITH TRACKBALL')

      CALL GSTT([illegible])

      RETURN
      END
```

END OF DATA

```
      SUBROUTINE USTMSG(PRCORE)
      IMPLICIT INTEGER(A-Z)

      CALL GSCH(1000,35)

      CALL GHLT

      GOTO(1,2,3,4,5),PRCORE

1     WRITE(15,1000)
1000  FORMAT('ADD')
      GOTO6

2     WRITE(15,2000)
2000  FORMAT('DELETE')
      GOTO6

3     WRITE(15,3000)
3000  FORMAT('RESUPPLY')
      GOTO6

4     WRITE(15,4000)
4000  FORMAT('COMBAT POSTURE')
      GOTO6

5     WRITE(15,5000)
5000  FORMAT('RELOCATE')

6     CALL GSTT(0,0)

      RETURN
      END
```

```
      SUBROUTINE WARNMG
      IMPLICIT INTEGER (A-Z)

      CALL GHLT

      CALL GSCH(1001,6)
      WRITE(15,1001)
1001  FORMAT('THIS ACTION WILL PREEMPT')

      CALL GSCH(1002,6)
      WRITE(15,1002)
1002  FORMAT('PLANNED UNIT ACTION')

      CALL GSTT(0,0)

      RETURN
      END
```

```
      SUBROUTINE CKINT(KEY)
      COMMON/ATT/IATT(12)
**
** RETURN A CODE CORRESPONDING TO THE INTERRUPT OCCURRING.
** RESETS IATT(1) TO ZERO AND RESTARTS THE DISPLAY WITH CALL GSTT(0,0).
**
** RETURNS -1 IF NO INTERRUPT HAS OCCURRED.
**
** FOR KEY PRESSED                   KEY=KEY+100 (100 TO 131)
** FOR KEY RELEASED                  KEY=KEY NUMBER (0 TO 31)
** FOR LIGHT PEN SWITCH PRESSED      KEY=134
** FOR LP SWITCH RELEASED            KEY=34
** FOR LP DETECT                     KEY=10000+ENTITY NUMBER
** FOR CYCLE TIMER                   KEY=40
** FOR JMP TO SUBPICTURE             KEY=42
** FOR JMP RETURN                    KEY=42
** FOR INTERRUPT WORD                KEY=46
** FOR RESET/HALT                    KEY=48
**
**
      KEY=-1
      IF (IATT(1).EQ.0) RETURN
10    IF (IATT(5).EQ.-1) ITEMP=100
      IF (IATT(5).EQ.0) ITEMP=0
20    IF (IATT(1).NE.34) GO TO 30
      KEY=ITEMP+34
      GO TO 1000
30    IF (IATT(1).NE.32) GO TO 40
      KEY=10000+IATT(3)
      GO TO 1000
40    IF (IATT(1).NE.30) GO TO 50
      KEY=40
      GO TO 1000
50    IF (IATT(1).NE.26) GO TO 60
      KEY=42
      GO TO 1000
60    IF (IATT(1).NE.24) GO TO 70
      KEY=44
      GO TO 1000
70    IF (IATT(1).NE.22) GO TO 80
      KEY=46
      GO TO 1000
80    IF (IATT(1).NE.20) GO TO 90
      KEY=48
      GO TO 1000
90    IF (IATT(1).NE.36) GO TO 100
      KEY=ITEMP+IATT(3)
      GO TO 1000
100   CONTINUE
1000  IATT(1)=0
      CALL GSTT(0,0)
      RETURN
      END
```

```
      SUBROUTINE LAMPS(ITOP,I2ND,I3RD,IBOTOM)

C  THIS ROUTINE TURNS OFF ALL PREVIOUSLY LIT KEYS, AND
C     TURNS ON ONLY THOSE SPECIFIED IN THE PARAMETER LIST

C  EACH VARIABLE IS TREATED AS A BINARY NUMBER FROM
C     0 TO 255.  FOR EACH NONZERO BIT, THE CORRESPONDING
C     LIGHT OF THE PROPER ROW WILL BE TURNED ON


      DIMENSION LAMP(4)
      LAMP(1)=IBOTOM
      LAMP(2)=I3RD
      LAMP(3)=I2ND
      LAMP(4)=ITOP
      CALL GLMP(1,-1,0)

      DO 30 I=1,4
      MAX=256
      DO 20 J=0,7
      MAX=MAX/2
      IF(LAMP(I) .LT. MAX) GO TO 10
      KEY=I*8-J-1
      CALL GLMP(1,KEY,1)
      LAMP(I)=LAMP(I)-MAX
10    IF(LAMP(I) .EQ. 0) GO TO 30
20    CONTINUE
30    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PENPIK(ENTITY)
      COMMON/BRANCH/DRTURN
      IMPLICIT INTEGER (A-C,E-Z)
      LOGICAL DRTURN

C  ****  ALLOWS USER TO SELECT A SCREEN DISPLAYED ENTITY USING THE
C  ****  LIGHTPEN.  THE ENTITY(S) ARE PREVIOUSLY MADE LIGHTPEN SENSITIVE.

      ENTITY=0
      ACCEPT=1
      RTURN=30
      DRTURN=.FALSE.

      CALL LAMPS(64,0,0,0)

1     CALL CKINT(KEY)
```

```
      IF(KEY.NE.RTURN)GOTO2
      IF(ENTITY.NE.0)CALL GPUT(3,130,3,0)
      CALL LAMPS(0,0,0,0)
      DRTURN=.TRUE.
      RETURN
2     IF(KEY.LE.10000)GOTO3
      IF(ENTITY.NE.0)CALL GPUT(3,130,3,0)
      ENTITY=KEY-10000
      CALL GENT(ENTITY)
      CALL GPUT(3,130,3,1)
      CALL LAMPS(64,0,0,2)
      GOTO1
3     IF(KEY.NE.ACCEPT.OR.ENTITY.EQ.0)GOTO1
      CALL GPUT(3,130,3,0)

      RETURN
      END
```

```
      SUBROUTINE DRWSGS(ENTITY,ARRAY,LENGTH)
      IMPLICIT INTEGER (A-Z)
      DIMENSION ARRAY(1)

C  ****  ROUTINE DRAWS A LIST OF COORDINATES CONTAINED IN ARRAY
C  ****  WITH LENGTH AS ENTITY

      ELEMNT=6
      CALL GBEG(ENTITY,ARRAY(1),ARRAY(2))
      CALL GPUT(5,1760,0,0)
      DO 1 J=1,LENGTH,2
      CALL GPUT(ELEMNT,53,ARRAY(J),ARRAY(J+1))
      ELEMNT=ELEMNT+1
1     CONTINUE

      RETURN
      END
```

```
      SUBROUTINE LPSENS(SWITCH,ENTITY,RANGE)
      IMPLICIT INTEGER (A-Z)

      DO 1   ENTNUM=ENTITY,RANGE
             CALL GENT(ENTNUM)
             CALL GPUT(3,130,4,SWITCH)
1     CONTINUE

      RETURN
      END
```

```
      SUBROUTINE COLOR(CODE)
      IMPLICIT INTEGER(A-Z)

C  ****  SETS COLOR VIA CONTROL REGISTER SET IN ELEMENT 4
C  ****  CODES:                              RED  0
C  ****                                      YELLOW  1
C  ****                                      ORANGE  2
C  ****                                      GREEN  3

      A=0
      B=0

      IF(CODE.GE.2)A=1
      IF(CODE.EQ.1.OR.CODE.EQ.3)B=1

      CALL GPUT(4,140,5,A)
      CALL GPUT(4,140,6,B)

      RETURN
      END
```

```
SUBROUTINE SHOCUR
IEL=5
CALL GPUT(IEL,1740,0,1)
IEL=IEL+1
CALL GPUT(IEL,1600,3,0)
IEL=IEL+1
CALL GPUT(IEL,1600,7,0)
RETURN
END
```